

Пей Ан

Сопряжение ПК







с внешними устройствами



Новые возможности персонального компьютера

Сетевые приложения и удаленный доступ

Игровой, параллельный и последовательный порты

Программы управления экспериментальными платами

Измерение аналоговых величин с помощью внешних устройств

Pei An

PC INTERFACING

Practical Guide to Centronic RS232 and Game Ports



Пей Ан

СОПРЯЖЕНИЕ ПК С ВНЕШНИМИ УСТРОЙСТВАМИ



Москва, 2001

ББК 32.973.26-02
А64

Ан П.

А64 Сопряжение ПК с внешними устройствами: Пер. с англ. – М.: ДМК Пресс, 2001. – 320 с.: ил.

ISBN 5-94074-076-6

Данная книга посвящена возможностям персонального IBM-совместимого компьютера по сопряжению с внешними устройствами через параллельный, последовательный и игровой порты, которые имеются практически в любом современном ПК. В качестве внешних устройств выступают ЦАП и АЦП, схемы управления электромоторами, трансиверы, модемы, различные индикаторы, датчики и пр.; приводятся тексты программ управления с подробными комментариями.

Книга предназначена для широкого круга читателей, интересующихся информатикой, электроникой и вычислительной техникой. Она будет полезна студентам технических вузов и колледжей в качестве учебного пособия при изучении аппаратной части ПК, а также радиолюбителям, которые стремятся наиболее полно использовать возможности домашнего компьютера. Начинающие программисты найдут здесь большое количество исходных текстов программ, а инженеры-электронщики почерпнут новые идеи для красивой реализации своих профессиональных проектов.

ББК 32.973.26-02

PC Interfacing, Practical Guide to Centronic RS232 and Game Ports by Pei An.

ISBN 0-24051-448-3 (англ.)

ISBN 5-94074-076-6 (рус.)

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Книга посвящена проблемам сопряжения персонального компьютера с современными электронными устройствами при помощи параллельных, последовательных и игровых портов. В ней приведено много примеров, показывающих, как ПК может собирать информацию из окружающего мира и управлять внешними устройствами. Кроме того, предлагается программное обеспечение, написанное на языках Turbo Pascal и Visual Basic. Это сочетание аппаратной и программной части и раскрывает суть понятия «сопряжение компьютера».

Наиболее известны параллельный, последовательный и игровой порты, которые встроены практически в каждый ПК. Поэтому схемы, рассмотренные в данной книге, можно использовать со всеми типами компьютеров: настольными, портативными, карманными IBM PC и совместимыми с ними, Macintosh, Amiga, PSION[1] и др.

Книга предназначена для широкого круга читателей, в числе которых:

- специалисты, использующие компьютер для взаимодействия с внешним миром;
- программисты, которые разрабатывают аналогичное ПО;
- инженеры, мечтающие соединить цифровые электронные устройства с ПК;
- студенты, желающие на практике усвоить, как компьютер сопрягается с внешними устройствами;
- все, кто изучает новейшие способы применения компьютеров.

Структурно книга разделена на восемь глав.

Глава 1 знакомит читателя с устройством параллельного, последовательного и игрового портов. В ней приводится необходимая техническая информация, рассказывается, как использовать программное обеспечение для управления портами.

В главе 2 речь идет о некоторых практических инструментах для проведения экспериментов по сопряжению компьютера. В частности, в ней описываются

---

[1] Программы, представленные в книге, не будут работать на компьютерах Macintosh, Amiga и PSION. – *Прим. науч. ред.*

конструкции экспериментальных плат для параллельного, последовательного и игрового портов. Платы обеспечивают визуальное отображение состояния контактов портов, что дает возможность проследить процессы ввода/вывода информации через порты. Экспериментальные платы использованы во всех опытах, описанных в книге.

В главе 3 приведено программное обеспечение для этих плат на языках программирования Turbo Pascal версии 6 для DOS (TP6), Turbo Pascal для Windows (TPW) и Visual Basic версии 3 (VB3). Предлагаемое ПО вы можете применять в собственных разработках.

В главе 4 излагаются основные методы расширения возможностей портов. Здесь приведены некоторые электрические схемы и примеры программ.

В главе 5 рассмотрены многочисленные способы управления внешними устройствами, в частности реле, светодиодами, двигателями постоянного тока, шаговыми двигателями, модулями визуального отображения информации, приборами, работающими от бытовой сети, и др. Даны электрические схемы и примеры программ.

Глава 6 посвящена вводу данных. Здесь обсуждаются вопросы, связанные с управлением аналого-цифровыми преобразователями, конверторами напряжения в частоту, различными датчиками. Экспериментальные схемы обеспечивают возможность считывания компьютером информации о температуре, скорости потока жидкости, интенсивности света, магнитных полях и т.д.

В главе 7 рассказывается, как соединить компьютер с другими устройствами, такими как цифро-аналоговые преобразователи, часы, модули памяти и генераторы сигналов.

Глава 8 посвящена вопросам удаленного доступа и сетевым приложениям. Здесь речь идет о модемах, радиоприемниках, радиопередатчиках и радиоретрансляторах.

Книга содержит большое количество практических схем и управляющих программ по сопряжению компьютера с внешними устройствами, изготовленными в основном компанией RS Components (http://www.rs-components.com/rs/). Фирменное обозначение компонентов, указанных в тексте, поможет читателям при проведении экспериментов. Облегчить работу призваны и листинги управляющих программ с комментариями. Программное обеспечение можно найти в Internet по адресу http://www.newnespress.com (архив ioexp.zip).

При написании книги были предприняты все необходимые проверки для того, чтобы обеспечить правильную и безопасную работу любой схемы или программы. Однако автор не несет ответственности за ошибки в принципиальных схемах и программах, которые могут функционировать некорректно и/или вызвать повреждение другого оборудования, к которому подсоединены экспериментальные устройства.

Будьте осторожны: некоторые устройства, представленные в этой книге, могут использовать опасные для человека напряжения. Соблюдайте правила техники безопасности.

Благодарности

Прежде всего хотелось бы поблагодарить мистера Дункана Энрайта (Duncan Enright) за идею написания этой книги. Также выражаю признательность доктору Шуишенгу Хе (Shuisheng He), доктору Йанкангу Ли (Jiankang Li), доктору Джинг Зао (Jing Zhao), доктору Фейбьао Зоу (Feibiao Zhou), доктору Ксиаохонгу Пенг (Xiaohong Peng) и доктору Синди Куи (Cindy Qiu) за то, что они прочитали рукопись. Кроме того, я очень благодарен за помощь в написании книги, а также за обеспечение примерами и соответствующей документацией следующим компаниям: RS Components, UCC International, Three Five Systems и Speak and Co. Ltd.

Используемые обозначения

Чтобы упростить восприятие материала, в книге приняты следующие обозначения:

- *Курсивом* в тексте выделены базовые термины и определения.
- `Моноширинным шрифтом` в книге набраны все листинги (фрагменты программного кода), выделены названия команд, адреса регистров, ячейки памяти ОЗУ.
- **Полужирным начертанием** отмечены названия элементов интерфейса (окон, пунктов меню, опций) при описании работы программ, а также кнопок некоторых электронных устройств.

Зарегистрированные торговые марки

В табл. 1 представлены сведения о фирмах-производителях электронного оборудования, упоминаемых на страницах этой книги.

Таблица 1. Фирмы-производители электронного оборудования и их торговые марки

Торговая марка	Компания
Amiga	Commodore Business Machines Corporation
Analog Devices	Analog Devices Inc.
Allegro MicroSystems	Allegro MicroSystems Inc.
Cystal Semiconductors	Cystal Semiconductors Inc.
Dallas Semiconductor	Dallas Semiconductor Corporation
GEC Plessey Semiconductors	GEC Plessey Semiconductors Ltd
Harris Semiconductors	Harris Corporation
Hewlett Packard	Hewlett Packard Corporation
Hitachi	Hitachi Ltd
Holtek	Holtek Microelectronics Inc.
IBM	International Business Machines

Таблица 1. Фирмы-производители электронного оборудования и их торговые марки (окончание)

Торговая марка	Компания
Isocom	Isocom Ltd
Maplin	Maplin plc
Maxim	Maxim Integrated Products Inc.
Microchip	Microchip Technology Inc.
MS-DOS, Visual Basic, Windows	Microsoft Corporation
National Semiconductors	National Semiconductors Inc.
NEC	NEC Corporation
Newport Components	Newport Components Inc.
Optek	Optek Technology Inc.
Philips Semiconductors	Philips Semiconductors
PSION	PSION plc
Quality Technologies	Quality Technologies
Radio Solutions	Radio Solutions Ltd
Radiometrix	Radiometrix Ltd
RS	RS Components Ltd
SGS-Thomson	SGS-Thomson Microelectronics
Siemens	Siemens AG
Sharp	Sharp Corporation
Speake & Co. Ltd	Speake & Co. Ltd
Texas Instruments	Texas Instruments Inc.
Three Five Systems	Three Five Systems Inc.
Timely	Timely Technology Ltd
Toshiba	Toshiba Corporation
Turbo Pascal	Borland International Inc.
UCC	UCC International Ltd
Xicor	Xicor Semiconductor Inc.

1. ПАРАЛЛЕЛЬНЫЙ, ПОСЛЕДОВАТЕЛЬНЫЙ И ИГРОВОЙ ПОРТЫ

Параллельный, последовательный и игровой порты – это наиболее распространенные порты ввода/вывода. В некоторых портативных компьютерах может не быть игрового порта, но параллельный и последовательный входят в стандартную комплектацию для всех типов ПК.

Изначально каждый из этих портов разрабатывался для определенного применения. Параллельные предназначались для соединения компьютеров с принтерами, последовательные – для подключения принтеров, модемов и мыши, а игровые – для присоединения джойстиков. Однако они могут использоваться и для других приложений, связанных с сопряжением компьютера с внешними устройствами. Периферийные устройства, созданные для этих портов, легко подключаются к IBM PC-совместимому компьютеру. Принципиальные схемы отличаются мобильностью и могут применяться для решения проблем сопряжения с любым оборудованием, которое оснащено указанными портами. Таким образом, полезно узнать, как они работают и каким образом обеспечивается наиболее эффективное их использование.

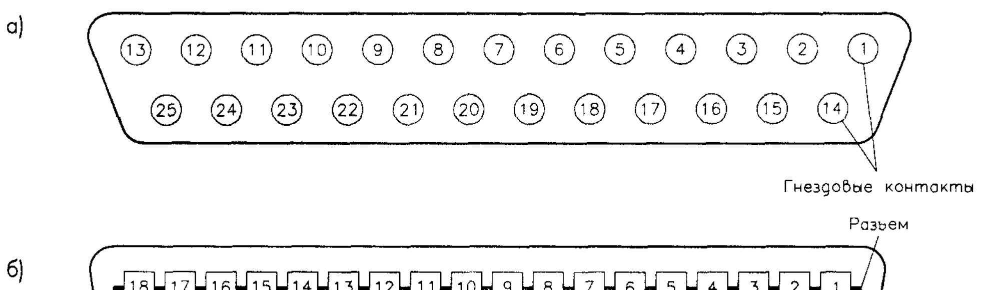
1.1. Параллельный порт

Порт Centronic, или *параллельный*, – это промышленный стандарт для подсоединения принтеров к компьютеру. Компьютер имеет по крайней мере один такой порт, встроенный в материнскую плату или представляющий собой отдельную интерфейсную карту ввода/вывода. Увеличить количество параллельных портов просто и недорого, можно установить четыре параллельных порта с логическими именами от LPT1 до LPT4. Команды управления принтером подробно не описываются.

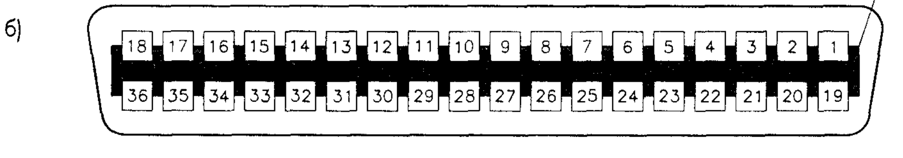
1.1.1. Разъемы

Разъемы порта для компьютера и принтера отличаются друг от друга. Первый – это 25-контактная розетка D-типа (рис. 1.1а), а второй – 36-контактная розетка параллельного типа (рис. 1.1б).

а)



б)



в)

Номера контактов на		Направление (относительно ПК)	Наименование	Назначение
компьютерах	принтерах			
1	1	Выход	STROBE	Строб данных
2	2	Выход	DB0	Бит данных 0
3	3	Выход	DB1	Бит данных 1
4	4	Выход	DB2	Бит данных 2
5	5	Выход	DB3	Бит данных 3
6	6	Выход	DB4	Бит данных 4
7	7	Выход	DB5	Бит данных 5
8	8	Выход	DB6	Бит данных 6
9	9	Выход	DB7	Бит данных 7
10	10	Вход	ACK	Подтверждение приема данных, готовность принтера
11	11	Вход	BUSY	Подтверждение занятости принтера
12	12	Вход	PE	Нет бумаги
13	13	Вход	SLCT	Принтер подключен к линии
14	14	Выход	LF/CR	Автоматический перевод строки после возврата каретки
15	32	Вход	ERROR	Ошибка в принтере
16	31	Выход	INITIALIZE	Установка параметров по умолчанию
17	36	Выход	SLIN	Выбор принтера
18 25	19 30, 33		GND	Витая пара, соединенная с "землей"
	18, 34		Не используется	
	16		LOGIC GND	Логическая 'земля'
	17		CHASSIS GND	Заземление на шасси

Рис. 1.1. Контакты на разъемах параллельного порта компьютера и принтера а – блочная часть 25-контактного гнездового разъема D-типа, вид со стороны задней стенки компьютера, б – блочная часть 36-контактного разъема параллельного типа, вид со стороны задней стенки принтера, в – назначение контактов разъемов параллельного порта

Назначение контактов обоих разъемов представлено на рис. 1.1в. Для соединения компьютера с принтером используется принтерный кабель (рис. 1.2) длиной не более 5 м.

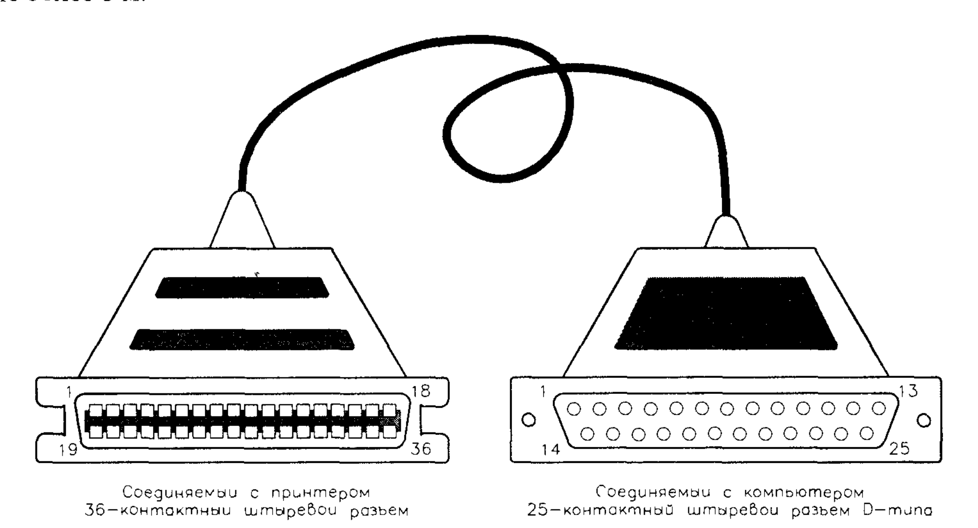


Рис. 1.2. Кабель принтера

1.1.2. Внутреннее устройство

Общая схема параллельного порта внутри ПК представлена на рис. 1.3. Восьмибитовые данные заносятся в DD1 во время записи в регистр с адресом базовый адрес + 0. Операция осуществляется командой WRITE_DATA.

Эти данные образуют группу. Они считываются компьютером из того же регистра через DD2 с помощью команды READ_DATA. Во время чтения выход DD1 должен иметь высокий уровень сопротивления, что достигается подачей на контакт 1 (выход разрешен) DD1 высокого уровня напряжения. Шестибитовое управляющее слово записывается в DD3 через регистр с адресом базовый адрес + 2 при помощи команды WRITE_CONTROL. Биты с 0 по 3 подаются на выход разъема и образуют группу управления. Некоторые биты инвертируются микросхемами с открытыми коллекторами на выходе (DD6 и DD7). Все выходные линии подключены к питанию +5 В через резисторы 4,7 кОм. Состояние этих линий считывается через регистр с адресом базовый адрес + 2 через DD4 посредством команды READ_CONTROL. Четвертый бит управляющего байта разрешает прерывание, а пятый бит открывает или закрывает выход DD1. Состояние пяти контактов разъема порта (группа состояния) компьютер считывает через DD4 с помощью команды READ_STATUS через регистр с адресом базовый адрес + 1. Входы линии подключены к питанию +5 В через резисторы 4,7 кОм, два входа инвертируются.

В первых конструкциях IBM PC контакт «выход разрешен» DD1 соединялся с «землей» для постоянного открывания выходов. Это была однонаправленная версия параллельного порта. Начиная с IBM PS/2, указанный контакт соединили

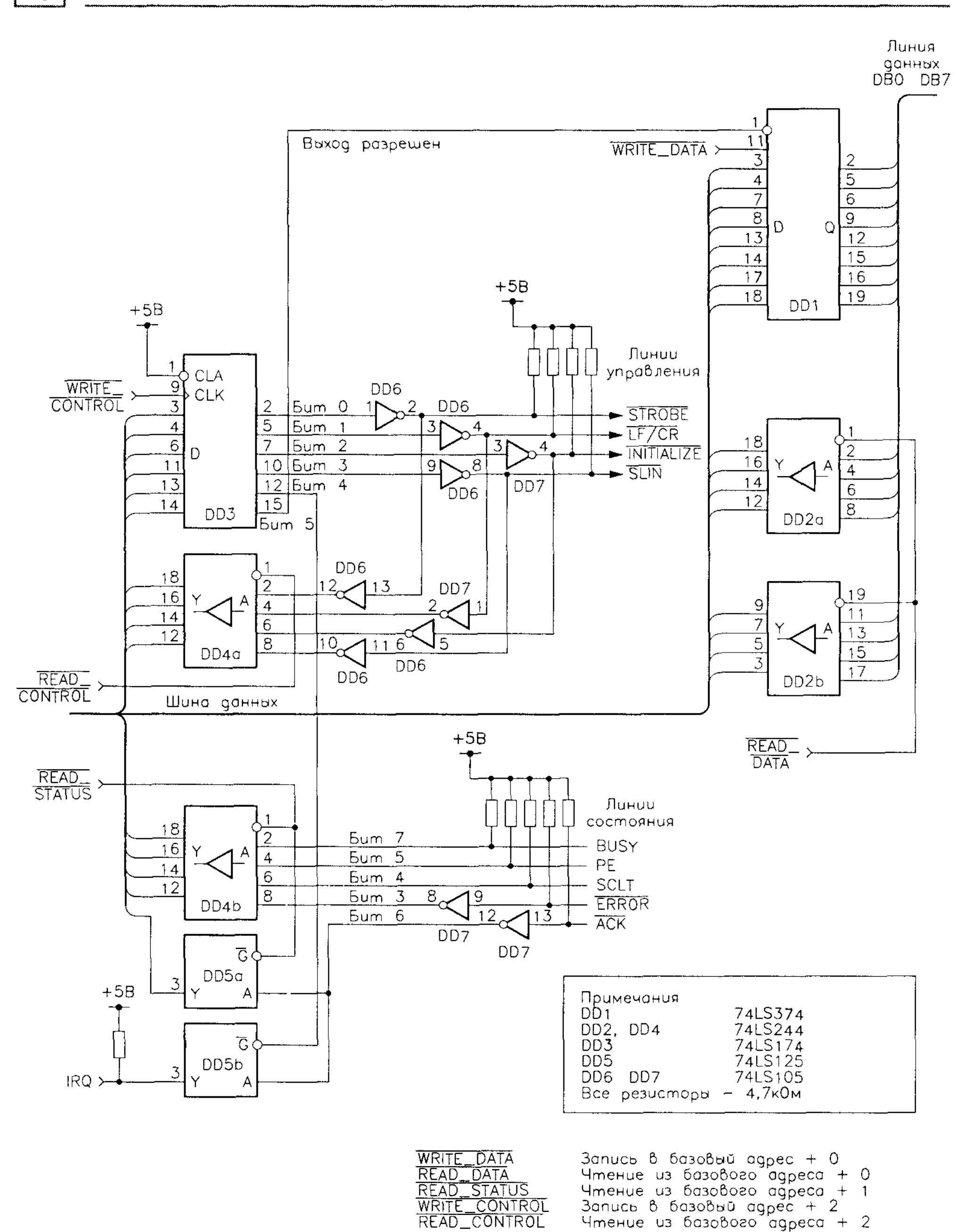


Рис. 1.3. Общая схема параллельного порта

с пятым битом регистра управления DD3 (см. рис. 1.3), и порт стал двунаправленным. Следует отметить, что многие параллельные порты, поставляемые со встроенными картами ввода/вывода, двунаправленные. Для любого контакта следует избегать короткого замыкания и/или соединения с шиной питания. Скорость передачи данных через параллельный порт превышает 1 Мб/с.

В этой главе детально рассматривается однонаправленный параллельный порт. Контакты порта образуют три группы: данных, управления и состояния. На рис. 1.4 представлена логическая структура параллельного порта.

Группа данных

Посылает данные от ПК на внешние устройства. Имеет восемь выходных линий и ассоциируется с байтом в адресном пространстве ввода/вывода процессоров x86. Адрес: базовый адрес.

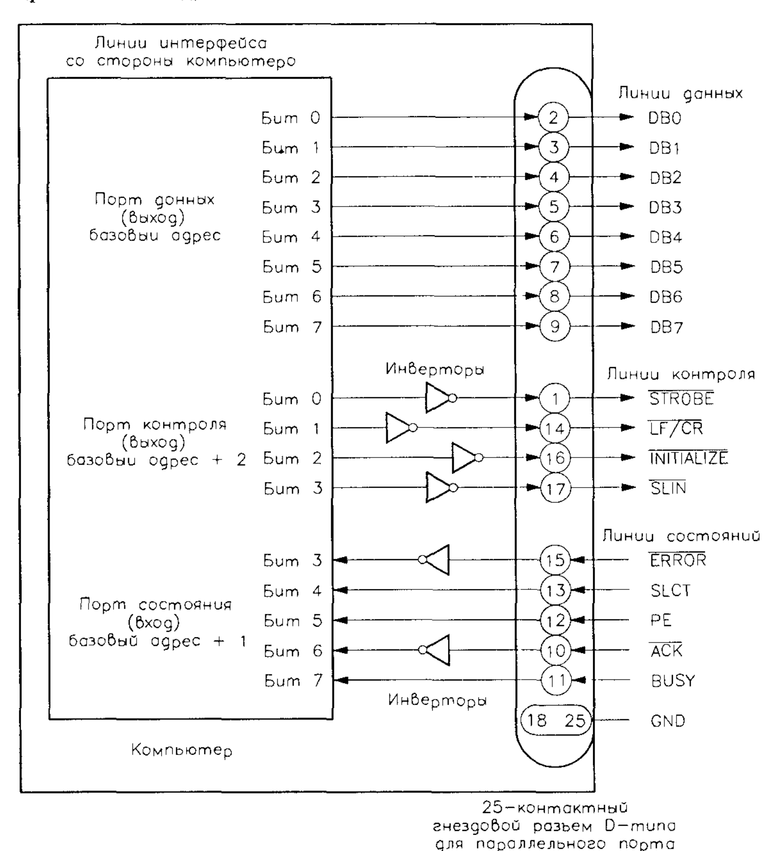


Рис. 1.4. Логическая структура параллельного порта

Группа управления

Контролирует внешнее устройство. Имеет четыре инвертированные выходные линии ($\overline{STROBE}$, $\overline{LF/CR}$, $\overline{SLIN}$ и $\overline{INITIALIZE}$). Адрес группы управления: базовый адрес + 2.

Группа состояния

Группа может использоваться компьютером для получения текущего состояния внешнего устройства, ее адрес: базовый адрес + 1. Имеет пять линий ($\overline{ERROR}$, SLCT, PE, $\overline{ACK}$ и BUSY). Линии $\overline{ERROR}$ и $\overline{ACK}$ инвертированы, остальные – нет.

Назначения регистров параллельного порта приведены в табл. 1.1.

Таблица 1.1. Назначения регистров параллельного порта

	Группа данных
Биты 0–7	Биты от 0 до 7, бит 0 – младший
	Группа управления
Бит 0 ($\overline{STROBE}$)	Строб-импульс данных Данные фиксируются по низкому уровню сигнала
Бит 1 ($\overline{LF/CR}$)	Автоматический перевод строки. При низком уровне принтер, получив символ CR (Carriage Return – возврат каретки), автоматически выполняет и функцию LF (Line Feed – перевод строки)
Бит 2 ($\overline{INITIALIZE}$)	Инициализация принтера. При низком уровне восстанавливаются параметры, принятые по умолчанию, каретка возвращается к началу строки
Бит 3 ($\overline{SLIN}$)	Выбор принтера. При высоком уровне принтер нечувствителен к остальным сигналам интерфейса
Бит 4 (IRQ)	Разрешение аппаратных прерываний. При высоком уровне разрешается прерывание по спаду сигнала на линии ($\overline{ACK}$) – сигнала запроса следующего байта
Бит 5 (Data I/O)	Управление направлением передачи (для PS/2 и выше). Запись единицы переводит порт в режим ввода. При чтении состояние бита не определено
Биты 6, 7	Не используются
	Группа состояния
Биты 0–2	Не используются
Бит 3 ($\overline{ERROR}$)	Ошибка. Низкий уровень свидетельствует о том, что бумага закончилась, о состоянии off-line или о внутренней ошибке принтера
Бит 4 (SLCT)	Выбор принтера. Высокий уровень показывает, что принтер включен и готов к работе
Бит 5 (PE)	Нет бумаги Высокий уровень означает, что бумага закончилась
Бит 6 ($\overline{ACK}$)	Импульс подтверждения приема байта (низким уровнем) и запрос на прием следующего. Может использоваться для формирования запроса на прерывание
Бит 7 ($\overline{BUSY}$)	Принтер занят Прием данных возможен только при низком уровне сигнала

Базовые адреса портов LPT1 и LPT2 следующие:

LPT1: 956 (3BCh) или 888 (378h)
LPT2: 632 (278h)

Базовый адрес для LPT1 зависит от конфигурации оборудования компьютера. Существует два способа получения базового адреса: проверка конфигурации оборудования вашего компьютера или нахождение адреса непосредственно из пользовательских программ с помощью средств, предлагаемых *базовой системой ввода/вывода* (BIOS) компьютера. При включении или перезагрузке компьютера BIOS проверяет наличие параллельных портов. Если они обнаруживаются, их базовые адреса (двухбайтовые слова) помещаются в определенные ячейки памяти ОЗУ. Для LPT1 это ячейки 0000h:0408h и 0000h:0409h. Первая содержит младший, вторая – старший байт адреса. Базовый адрес LPT1 можно получить, считав содержимое этих ячеек. Ячейки памяти для портов LPT1 – LPT4 приведены ниже:

LPT1: 0000:0408h – 0000:0409h
LPT2: 0000:040Ah – 0000:040Bh
LPT3: 0000:040Ch – 0000:040Dh
LPT4: 0000:040Eh – 0000:040Fh

Кроме того, используется еще одна ячейка памяти: 0000:4011h. Она содержит сведения об общем количестве параллельных портов, установленных на компьютере. Эта информация хранится в битах 6 и 7:

бит 7 = 0, бит 6 = 0	параллельные порты не установлены
бит 7 = 0, бит 6 = 1	установлен один параллельный порт
бит 7 = 1, бит 6 = 0	установлено два параллельных порта
бит 7 = 1, бит 6 = 1	установлено три параллельных порта

1.1.3. Программное управление

В данном разделе приводится информация, необходимая для программирования параллельного порта, и даются начальные сведения по логическим операциям над битами.

Получение базового адреса параллельного порта

Следующая программа написана на языке QBASIC. Она выводит общее количество параллельных портов и их базовые адреса от LPT1 до LPT3. Строка 20 считывает байт, находящийся в ячейке памяти 0000:0411h, используя команду PEEK(). Биты 7 и 6 выделяются с помощью маски AND (128 + 64) . Затем результат сдвигается на 6 бит по направлению к младшему разряду с помощью деления на 64. Строка 30 считывает два байта из двух ячеек памяти, содержащих младший и старший байты базового адреса LPT1. Строки 40 и 50 делают то же самое для LPT2 и LPT3.

```
10 DEF SEG = 0
20 PRINT "Number of Centronic ports:", (PEEK(&H411) AND (128 + 64))/64
30 PRINT "Address of LPT1:", PEEK(&H408)+256*PEEK(&H409)
40 PRINT "Address of LPT2:", PEEK(&H40A)+256*PEEK(&H40B)
50 PRINT "Address of LPT3:", PEEK(&H40C)+256*PEEK(&H40D)
60 INPUT x
```

Следующая процедура на языке TP6 определяет количество установленных параллельных портов и присваивает полученное значение переменной `Number_of_LPT`. Затем она считывает их базовые адреса из ячеек памяти, где хранятся адреса портов LPT1 – LPT4. Далее программа предлагает указать тот LPT-порт, к которому будет присоединено внешнее устройство. И наконец, она присваивает выбранный базовый адрес переменной `Centronic_address`. В Turbo Pascal 6 для считывания содержимого ячеек памяти используются функции `mem(основание: смещение)` и `memw(основание: смещение)`. Функция `mem(...)` считывает байт из ячейки памяти, а `memw(...)` – двухбайтовое слово из указанной ячейки памяти и одной ячейки выше.

```
(*-Библиотека ресурсов № А1 (определение базовых адресов LPT-портов)-.*)
Procedure Centronic_Address
(* $000:$0408 содержит базовый адрес для LPT1,
   $000:$040A содержит базовый адрес для LPT2,
   $000:$040C содержит базовый адрес для LPT3,
   $000:$040e содержит базовый адрес для LPT4,
   $000:$0411 содержит количество параллельных портов.*)
var
   lpt:array[1..4] of integer;
   number_of_lpt,LPT_number,code:integer;
   kbchar:char;
begin
      clrscr;
      LPT_number:=1;                    (*Для установки принтера по умолчанию.*)
      number_of_lpt:=mem($0000:$0411);  (*Считывает количество установленных
                                        параллельных портов.*)
      number_of_lpt:=(number_of_lpt and (128+64)) shr 6;     (*Манипуляции с битами.*)
      lpt[1]:=memw($0000:$0408);        (*Процедура считывания из памяти.*)
      lpt[2]:=memw($0000:$040A);
      lpt[3]:=memw($0000:$040C);
      lpt[4]:=memw($0000:$040e);
      textbackground(blue); clrscr;
      textcolor(yellow); textbackground(red); window(10,22,70,24); clrscr;
      writeln('Number of LPT installed:',number_of_lpt:2);
      writeln('Addresses for LPT1 to LPT4: ', lpt[1]:3,' ',lpt[2]:3,'  ',lpt[3]:3,' ',lpt[4]:3);
      write('Select LPT to be used(1,2,3,4): ');
      delay(1000);
      if number_of_lpt>1 then {Выбор порта, если установлено несколько портов.}
      begin
         repeat
            kbchar:=readkey;            (*Считывание значения нажатой клавиши.*)
            val(kbchar,LPT_number,code); (*Преобразование символа в число.*)
         until (LPT_number>=1) and (LPT_number<=4) and (lpt[LPT_number]<>0);
      end;
      clrscr;
      P_address:=lpt[LPT_Number];
```

```
        writeln('Your selected printer interface: LPT',LPT_number:1);
        write('LPT address: ',P_address:3);
        delay(1000);
        textbackground(black); window(1,1,80,25); clrscr;
  end;
```

Функция centronic(x) написана на языке Turbo Pascal для Windows. Она может быть вызвана программой, написанной на другом языке программирования для Windows, например Visual Basic или Visual C, если ее оформить в виде библиотеки динамической компоновки DLL. Centronic(0) возвращает количество установленных LPT-портов, Centronic(1) – базовый адрес LPT1, Centronic(2) – базовый адрес LPT2 и т.д.

```
Function Centronic(x:integer):integer; export;
(* $000:$0408 содержит базовый адрес для LPT1,
   $000:$040A содержит базовый адрес для LPT2,
   $000:$040C содержит базовый адрес для LPT3,
   $000:$040e содержит базовый адрес для LPT4,
   $000:$0411 содержит количество параллельных портов.*)
var
   number_of_LPT,LPT1,LPT2,LPT3,LPT4:integer;
   lpt1,lpt2,lpt3,lpt4: integer;
begin
   number_of_LPT:=mem($40:$11);     (*Считывает количество LPT-портов.*)
   number_of_LPT:=( number_of_LPT and (128+64)) shr 6;
   lpt1:=0; lpt2:=0; lpt3:=0; lpt4:=0;
   lpt1:=memw($40:$08);             (*Процедура считывания из памяти.*)
   lpt2:=memw($40:$0A);
   lpt3:=memw($40:$0C);
   lpt4:=memw($40:$0e);
   case x of
       0: centronic:=Number_of_LPT;
       1: Centronic:=lpt1;
       2: Centronic:=lpt2;
       3: Centronic:=lpt3;
       4: Centronic:=lpt4;
   end;
end;
```

Ввод/вывод данных через параллельный порт

Существует несколько способов записи информации в параллельный порт.

Команды принтера и процедуры прерываний BIOS

В QBASIC команда вывода на печать – PRINT, в TP6 – writeln(lst). Другой способ управления принтером заключается в использовании прерывания BIOS – INT 17h. Временные диаграммы вывода данных через параллельный порт показаны на рис. 1.5.

Сначала компьютер проверяет, готов ли принтер к приему новых данных. Для этого необходимо проконтролировать состояние линии BUSY. Когда на ней низкий уровень («не занят»), ПК записывает данные в регистр данных. Через 500 нс компьютер переводит сигнал $\overline{\text{STROBE}}$ в низкий уровень, который, в свою очередь,

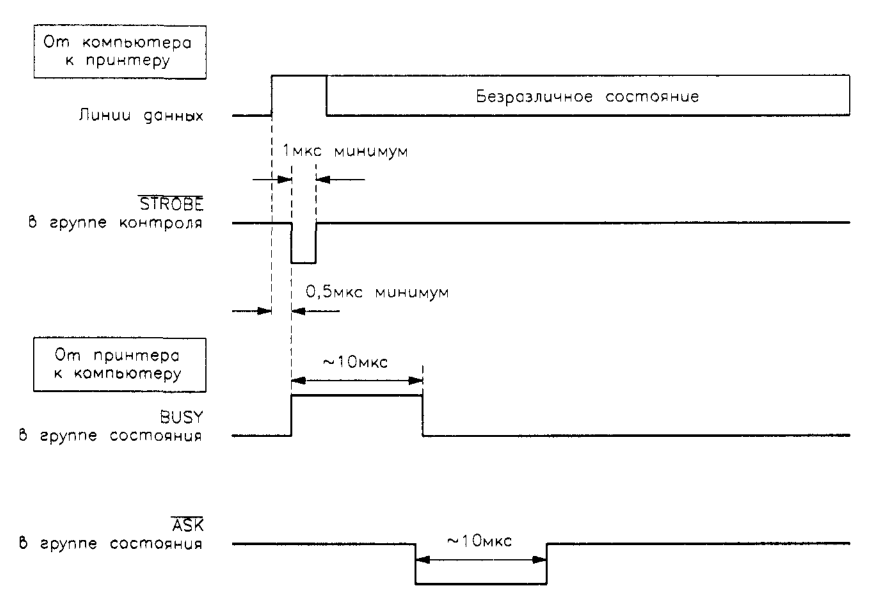


Рис. 1.5. Временные диаграммы взаимодействия компьютера с принтером

переводит принтер в состояние «занят» (BUSY=1). Принтер принимает и обрабатывает данные, а затем переводит сигнал $\overline{\text{ACK}}$ в низкий уровень для индикации, что принятые данные обработаны. В то же время принтер переводит линию BUSY в низкий уровень.

Практически в каждом языке программирования существуют инструкции по управлению принтером. Однако нужно учитывать, что этот метод недостаточно гибок для операций ввода/вывода при сопряжении ПК с внешними устройствами.

Если к компьютеру подсоединена внешняя схема, то в ней должна быть специальная схема для генерации сигналов BUSY и $\overline{\text{ACK}}$. Удобнее всего, если ПК при взаимодействии будет использовать только линии $\overline{\text{ACK}}$. BUSY постоянно соединяется с цифровой «землей» для индикации, что схема готова к приему данных, а PE показывает, что в принтере есть бумага; на линию $\overline{\text{ERROR}}$ подается высокий уровень. Если не соединить таким образом линии PE и $\overline{\text{ERROR}}$, при запросе компьютера на печать будут выдаваться сообщения об ошибке. Более гибкий способ управления параллельным портом – непосредственный доступ к регистрам порта.

Непосредственный доступ к регистрам порта

Это метод управления портом при помощи непосредственного доступа к его регистрам. Параллельный порт рассматривается как три отдельных регистра ввода/вывода, два из которых предназначены для вывода данных, а один – для ввода.

Рассмотрим пример управления портом LPT1. Так как адреса регистров данных, управления и состояния имеют номера 888, 890 и 889 соответственно, для

записи информации в регистры данных и управления применяются следующие команды языка QBASIC:

```
OUT 888, X
OUT 890, X
```

где X – записываемое значение в десятичном представлении. Некоторые линии порта управления инвертированы, что необходимо учитывать при выводе данных. Для чтения данных из регистра состояния можно пользоваться следующей командой:

```
Y=INP(889)
```

где Y – десятичное входное значение. Биты входных данных соответствуют номерам с 3 по 7 регистра состояния, некоторые из них инверсные.

Следующие процедуры на языке TP6 записывают информацию в регистры данных и управления. Обеим процедурам требуются базовый адрес выбранного параллельного порта и значение выходных данных. Выходные данные для регистра управления предусматривают проведение некоторых преобразований над битами.

```
(*-Библиотека ресурсов № A4 (запись информации в порт данных компьютера)-.*)
Procedure Write_data_port(P_address:integer, port_data:byte);
(*Биты регистра данных не инвертированы.*)
begin
  port(P_address):=port_data;  (*Ввод байта данных в регистр данных.*)
end;

(*-Библиотека ресурсов № A5 (запись данных в регистр управления.)-.*)
Procedure Write_control_port(P_address:integer; port_data:byte);
(*Биты 0, 1 и 3 инвертированы. Требуются преобразования над битами.*)
begin
  if port_data and 1=1 then port_data:=port_data and (255-1)
      else port_data:=port_data or 1;
  if port_data and 2=2 then port_data:=port_data and (255-2)
      else port_data:=port_data or 2;
  if port_data and 8=8 then port_data:=port_data and (255-8)
      else port_data:=port_data or 8;
  port(P_address+2):=port_data;    (*Ввод байта данных в регистр управления.*)
end;
```

Следующая функция на языке TP6 считывает биты с 3 по 6 из регистра состояния. Она требует базовый адрес выбранного параллельного порта. Функция также выполняет битовые преобразования и возвращает значение четырехбитовых входных данных.

```
(*-Библиотека ресурсов № A3 (считывание данных в компьютер).-*)
Function Read_status_port(P_address:integer):byte;
var
  byte1:byte;
begin
  byte1:=port(P_address+1);        (*Считывание байта из регистра состояния.*)
  byte1:=byte1 and 120;            (*01111000 (от старшего к младшему) and 0dddd... = 0dddd000.*)
  Read_status_port:=byte1 shr 3;   (*Сдвиг на 3 бита вправо, Read_status_port=0000hhhh.*)
end;
```

В следующих примерах, написанных на языке Turbo Pascal для Windows, представлен ввод и вывод данных через параллельный порт.

```
(*-Библиотека ресурсов № А4 (запись информации в регистр данных).-*)
Function Write_data_port(P_address:integer, port_data:integer); export;
(*Биты регистра данных не инвертированы.*)
begin
  port(P_address):=port_data      (*Ввод байта в регистр данных.*)
end;

(*-Библиотека ресурсов № А5 (запись данных в регистр управления).-*)
Function Write_control_port(P_address:integer; port_data:integer) integer; export;
(*Биты 0, 1 и 3 инвертированы Требуются преобразования над битами.*)
begin
  if port_data and 1=1 then port_data:=port_data and (255-1)
     else port_data:=port_data or 1;
  if port_data and 2=2 then port_data:=port_data and (255-2)
     else port_data:=port_data or 2;
  if port_data and 8=8 then port_data:=port_data and (255-8)
     else port_data:=port_data or 8,
  port(P_address+2):=port_data;    (*Ввод байта данных в регистр управления.*)
end,

(*-Библиотека ресурсов № А3 (считывание данных в компьютер).-*)
Function Read_status_port(P_address:integer).integer; export;
var
  byte1 byte;
begin
  byte1 =port(P_address+1);        (*Считывание байта из регистра состояния *)
  byte1 =byte1 and 120;            (*01111000 (от старшего к младшему) and 0dddd  = 0dddd000 *)
  Read_status_port:=byte1 shr 3;   (*Сдвиг на 3 бита вправо, Read_status_port=0000hhhh.*)
end;
```

Преобразования над битами

В данном разделе рассматриваются некоторые основные приемы битовых преобразований. Рассказывается о том, что такое вес бита, как сделать определенный бит единицей или нулем, описываются процедуры побитовых сдвигов.

Вес бита

Соотношения между битами и их весами приведены ниже:

бит 0	1 (десятичное значение)
бит 1	2
бит 2	4
бит 3	8
бит 4	16
бит 5	32
бит 6	64
бит 7	128

Присвоение биту единичного значения

Следующий пример демонстрирует, как сделать бит 3 (вес равен 8) регистра данных единицей (независимо от его исходного значения), оставив другие биты неизменными:

```
10 X=Original_data OR 8
20 OUT 888, X
```

Строка 10 выполняет операцию логического сложения (OR). Таблица истинности этой операции приведена ниже:

```
0 OR 0 = 0
0 OR 1 = 1
1 OR 0 = 1
1 OR 1 = 1
```

Пример поразрядной операции OR

Данные-1:	XXXXXXXX (биты от 7 до 0)
Данные-2:	00001000
Данные-1 OR Данные-2:	XXXX1XXX

Присвоение биту нулевого значения

Следующий пример на языке QBASIC показывает, как сделать бит 4 (вес равен 16) регистра данных нулевым:

```
10 X=Original_data and (255-16)
20 OUT 888, X
```

Строка 10 выполняет операцию логического умножения (AND). Таблица истинности этой операции выглядит следующим образом:

```
0 AND 0 = 0
0 AND 1 = 0
1 AND 0 = 0
1 AND 1 = 1
```

Пример поразрядной операции AND

Данные-1:	XXXXXXXX (биты от 7 до 0)
Данные-2:	11101111
Данные-1 AND Данные-2:	XXX0XXXX

Сдвиг битов вправо или влево

Уже говорилось, что при чтении данных из регистра состояния информативными являются только биты 3–6. Чтобы их выделить, необходимо произвести побитовый сдвиг. В TP6 для этих целей существует две процедуры: SHL – сдвиг битов влево (к старшему разряду) и SHR – сдвиг вправо (к младшему разряду). Следующий пример демонстрирует выполнение обеих операций:

Данные:	11111111 (биты с 7 по 0)
255 SHL 3:	11111000
255 SHR 3:	00011111

1.2. Последовательный интерфейс RS232

Последовательный интерфейс RS232 – это промышленный стандарт для последовательной двунаправленной асинхронной передачи данных. Он используется в компьютерах при подсоединении принтеров, модемов, мыши и т.д. Максимальное расстояние, позволяющее организовать связь, равно 20 м.

В отличие от параллельного порта, состоящего из восьми информационных линий и за один такт передающего байт, порт RS232 требует наличия только одной такой линии, по которой последовательно передается бит за битом. Это позволяет сократить количество информационных линий для передачи данных между устройствами, но уменьшает скорость.

1.2.1. Последовательная передача данных

Последовательный поток данных состоит из битов синхронизации и собственно битов данных. Формат последовательных данных содержит четыре части: стартовый бит, биты данных (5–8 бит), проверочный и стоповый биты; вся эта конструкция иногда называется *символом*. На рис. 1.6 изображен типичный формат последовательных данных.

Когда данные не передаются, на линии устанавливается уровень логической единицы. Это называется режимом ожидания. Начало режима передачи данных характеризуется передачей уровня логического нуля длительностью в одну элементарную посылку. Такой бит называется *стартовым*. Биты данных посылаются последовательно, причем младший бит – первым; всего их может быть от пяти

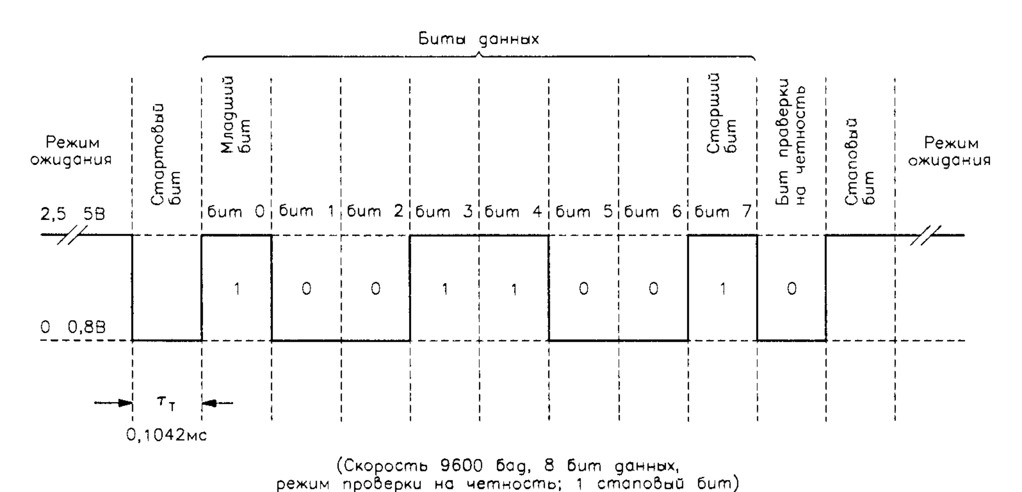


Рис. 1.6. Формат последовательных данных, формируемых UART

до восьми. За битами данных следует проверочный бит, предназначенный для обнаружения ошибок, которые возникают во время обмена данными. Последней передается стоповая посылка, информирующая об окончании символа. Стоповый бит передается уровнем логической единицы. Длительность стоповой посылки – 1, 1,5 или 2 бита. Специально разработанное электронное устройство, генерирующее и принимающее последовательные данные, называется *универсальным асинхронным приемопередатчиком* (Universal Asynchronous Receiver Transmitter, UART).

Обмен информацией с помощью микросхем UART происходит следующим образом. Приемник обнаруживает первый фронт стартового бита и выжидает один или полтора тактовых интервала, поскольку считывание должно начаться точно в середине первой посылки. Через один тактовый интервал считывается второй бит данных, причем это происходит точно в середине второй посылки. После окончания информационного обмена приемник считывает проверочный бит для обнаружения ошибок и стоповый бит, а затем переходит в режим ожидания следующей порции данных.

Скорость передачи информации в последовательном интерфейсе измеряется в *бодах* (бод – количество передаваемых битов за 1 с). Стандартные скорости равны 110, 150, 300, 600, 1200, 2400, 4800, 9600 и 19200 бод. Зная скорость в бодах, можно вычислить число передаваемых символов в секунду. Например, если имеется восемь бит данных без проверки на четность и один стоповый бит, то общая длина последовательности, включая стартовый бит, равна 10. Скорость передачи символов соответствует скорости в бодах, деленной на 10. Таким образом, при скорости 9600 бод (см. рис. 1.6) будет передаваться 960 символов в секунду.

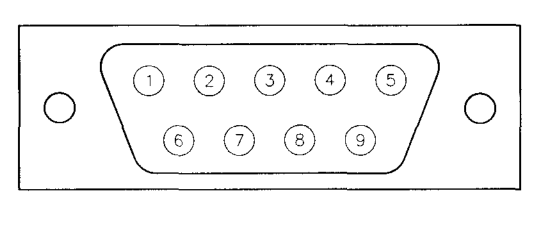
Проверочный бит предназначен для обнаружения ошибок в передаваемых битах данных. Когда он присутствует, осуществляется проверка на четность или нечетность. Если интерфейс настроен на проверку по четности, такой бит будет выставляться в единицу при нечетном количестве единиц в битах данных, и наоборот. Это простейший способ проверки на наличие одиночных ошибок в передаваемом блоке данных. Однако, если во время передачи искажению подверглись несколько битов, подобная ошибка не обнаруживается. Проверочный бит генерируется передающим UART таким образом, чтобы общее количество единиц было нечетным или четным числом в зависимости от настройки интерфейса; приемное устройство должно иметь такую же настройку. Приемный UART считает количество единиц в принятых данных. Если данные не проходят проверку, генерируется сигнал ошибки.

Большинство компьютеров, совместимых с IBM PC, использует UART 16450, с IBM PC XT – UART 8250. В UART применяются уровни напряжения ТТЛ. Для передачи данных по каналу связи напряжение с помощью специализированных преобразователей конвертируется с инверсией: логическому нулю соответствует диапазон напряжений от +3 до +12 В, логической единице – от –3 до –12 В.

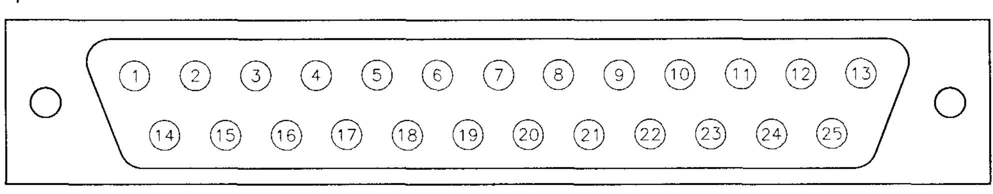
1.2.2. Разъем и кабель порта RS232

Стандартный последовательный порт имеет 25- или 9-контактный разъем. На рис. 1.7 приведены назначения контактов этих разъемов.





б)



в)

25 контактов	9 контактов	Наименование	Направление (относительно ПК)	Описание
1		PROT		Защитное заземление
2	3	TD	Выход.	Передаваемые данные
3	2	RD	Вход	Принимаемые данные
4	7	RTS	Выход	Запрос на передачу
5	8	CTS	Вход	Очищен для передачи
6	6	DSR	Вход	Готовность внешнего устройства
7	5	GND		Сигнальное заземление
8	1	DCD	Вход	Обнаружение информационного сигнала
20	4	DTR	Выход	Компьютер к обмену данными готов
22	9	RI	Вход	Индикатор звонка
23		DSRD	Вход/выход	Детектор скорости передачи данных

Рис. 1.7. Функции контактов разъемов RS232 на компьютере а – блочная часть 9-контактного штыревого разъема, вид со стороны задней стенки компьютера, б – блочная часть 25-контактного штыревого разъема, вид со стороны задней стенки компьютера, в – назначение контактов разъемов последовательного порта

В табл. 1.2 указано назначение сигналов последовательного интерфейса.

На рис. 1.8 представлены два типа соединений между компьютером и внешним устройством по протоколу RS232. Стрелки показывают направление потоков данных. На рис. 1.8а представлено так называемое *нуль-модемное соединение*. На рис. 1.8б изображено соединение, использующее только три линии: первая – для передачи данных, вторая – для приема, третья – общая. Соединение организовано таким образом, что передаваемые данные от первого устройства поступают на приемную линию второго.

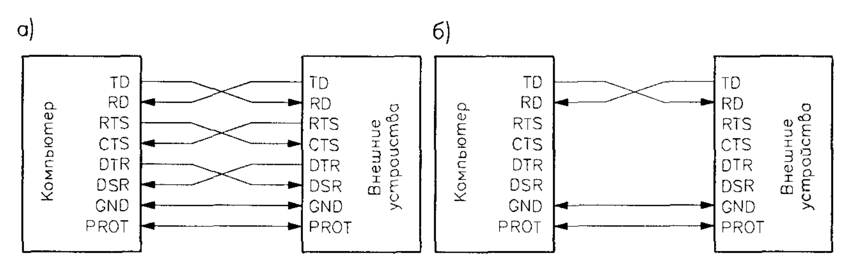


Рис. 1.8. Соединение компьютера и внешнего устройства по протоколу RS232 а – с использованием нуль-модемного кабеля, б – при помощи трех линий

Таблица 1.2. Назначение сигналов последовательного интерфейса

PROT	Защитное заземление Соединяется с металлическим экраном кабеля и корпусом оборудования
GND	Линия заземления Общий провод для всех сигналов
TD	Передаваемые данные Последовательные данные передаются компьютером по этой линии
RD	Принимаемые данные Последовательные данные принимаются компьютером по этой линии
RTS	Запрос на передачу Линия взаимодействия, которая показывает, что компьютер готов к приему данных Линия управляется со стороны компьютера, если взаимодействия не требуется, она может использоваться как двоичный выход
CTS	Готовность к передаче Линия взаимодействия, с помощью которой внешнее устройство сообщает компьютеру, что оно готово к передаче данных Если взаимодействия не требуется, она может использоваться как двоичный вход
DTR	Компьютер готов Линия взаимодействия показывает, что компьютер включен и готов к связи Линия управляется со стороны компьютера, если взаимодействие не требуется, она может использоваться как двоичный выход
DSR	Готовность внешнего устройства Линия взаимодействия, с помощью которой внешнее устройство сообщает компьютеру, что оно включено и готово к связи Если взаимодействия не требуется, она может использоваться как двоичный вход

1.2.3. Внутреннее аппаратное устройство

Компьютер, совместимый с IBM PC, может иметь до четырех последовательных портов. Они маркируются как COM1 – COM4. Каждый COM-порт формируется отдельным UART 16450, установленным внутри компьютера.

UART 8250/16450

На рис. 1.9 показано внутреннее устройство UART. В нем имеются восемь восьмибитовых регистров. Адреса ввода/вывода этих регистров вычисляются добавлением смещения регистра к базовому адресу COM-порта. Смещения и функции регистров UART таковы:

- 00h – буферный регистр передатчика/буферный регистр приемника: используется для обмена данными

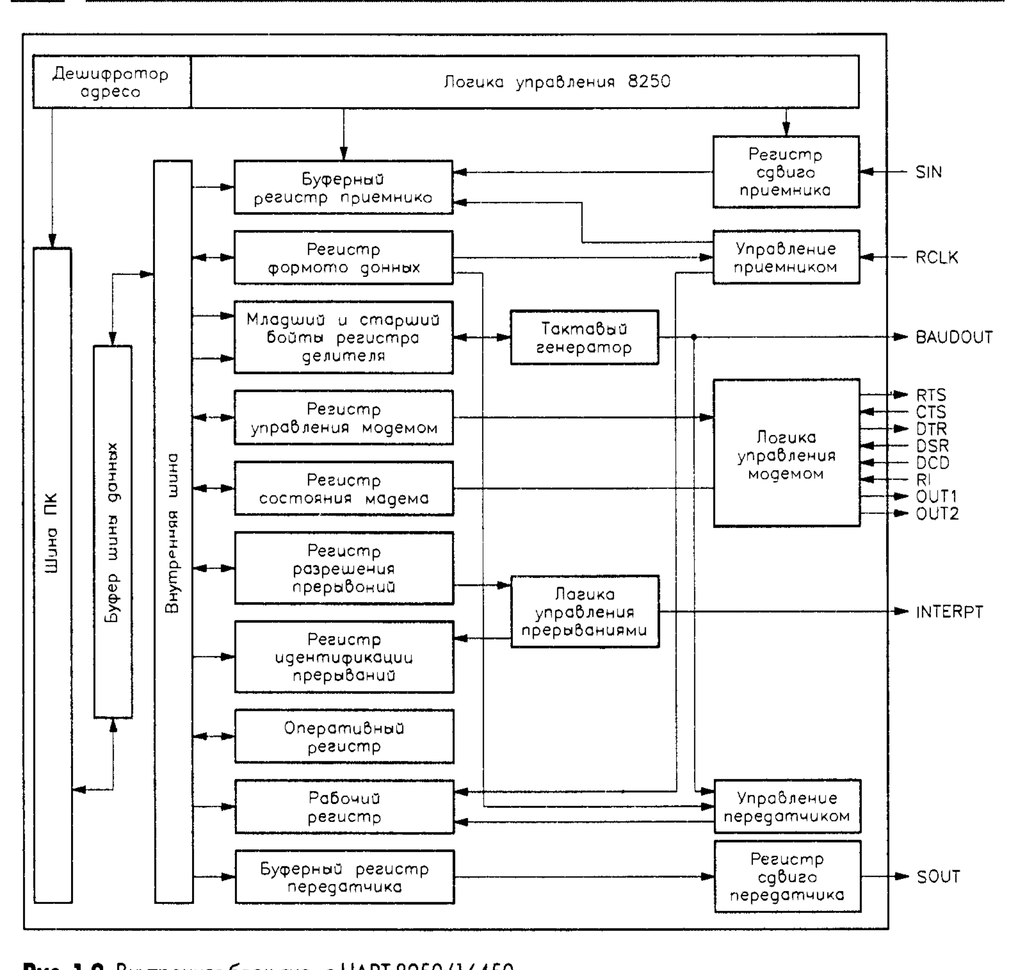


Рис. 1.9. Внутренняя блок-схема UART 8250/16450

- 01h – регистр разрешения прерываний: устанавливает режим запроса прерываний
- 02h – регистр идентификации прерываний: проверяет режим запроса прерываний
- 03h – регистр формата данных: устанавливает формат последовательных данных
- 04h – регистр управления модемом: устанавливает управление модемом (RTS, DTR и т.д.)
- 05h – регистр состояния приемопередатчика: содержит информацию о состоянии приемника и передатчика
- 06h – регистр состояния модема: содержит текущее состояние линий DCD, RI, DSR и CTS
- 07h – регистр сверхоперативной памяти: работает как байт памяти

Смещение 00h указывает на буферный регистр чтения приемника и регистр записи передатчика, который доступен, когда бит DLAB в регистре формата данных

(смещение 03h) равен нулю. Если по этому адресу записан байт, то он передается в регистр сдвига передатчика и последовательно поступает на выход. Во время приема происходит обратная операция: после того как данные успешно приняты и с помощью регистра сдвига преобразованы в параллельный формат, они передаются в буферный регистр приемника. Когда информация из этого регистра считана, он очищается и готов к приему следующего блока данных.

По смещению 01h от базового находится регистр разрешения прерываний, посредством которого можно настраивать прерывания, генерируемые UART. Назначения битов этого регистра приведены ниже:

```
0 0 0 0 SINP ERBK TBE RxRD
```

биты с 7 по 4	всегда нули
SINP	1 = прерывание по изменению состояния линий CTS, DSR, DCD и RI 0 = нет прерывания
ERBK	1 = прерывание при ошибке приема данных 0 = нет прерывания
TBE	1 = прерывание, когда регистр передатчика пуст 0 = нет прерывания
RxRD	1 = прерывание при получении данных 0 = нет прерывания

По смещению 02h находится регистр идентификации прерываний. При возникновении прерывания нулевой бит этого регистра устанавливается в 0. Биты 1 и 2 указывают причину прерывания. Биты с 7 по 3 не используются и всегда равны нулю. Назначение битов регистра следующее:

```
0 0 0 0 0 ID1 ID0 PND
```

PND	1 = нет прерывания 0 = прерывание
ID1, ID0	00 = изменение входного сигнала RS232 (приоритет 3) 01 = регистр передатчика пуст (приоритет 2) 10 = в буферном регистре приемника данные готовы (приоритет 1) 11 = ошибка передачи данных или остановка (приоритет 0, высшая степень)

Если процесс обмена данными организован по прерываниям, то установившееся прерывание должно быть сброшено; в противном случае корректность обмена данными нарушится. Действия, необходимые для очистки прерывания, таковы:

ID1 = 0, ID0 = 0	чтение содержимого регистра состояния модема (06h)
ID1 = 0, ID0 = 1	запись в регистр передатчика (00h) или чтение регистра идентификации прерываний (02h)
ID1 = 1, ID0 = 0	чтение байта данных из буферного регистра приемника (00h)
ID1 = 1, ID0 = 1	чтение регистра состояния приемопередатчика (05h)

По смещению 03h находится регистр формата данных, который определяет такие параметры передаваемых данных, как скорость, количество битов данных, количество стоповых битов и настройка проверочного бита. Назначение битов регистра приведено ниже:

DLAB BRK PAR2 PAR1 PAR0 STOP DAB1 DAB0

DLAB	1 = доступ к установке скорости
	0 = доступ к регистру приемника/регистру передатчика (00h) и к регистру разрешения прерываний (01h)
BRK	1 = остановка включена
	0 = остановка выключена
PAR2,1,0	000 = нет проверки
	001 = нечетная
	011 = четная
	101 = всегда 1
	111 = всегда 0
STOP	1 = 2 стоповых бита
	0 = 1 стоповый бит
DAB1,0	00 = 5 бит данных
	01 = 6 бит данных
	10 = 7 бит данных
	11 = 8 бит данных

Когда бит DLAB равен 1, регистры приемопередатчика (00h) и разрешения прерываний (01h) используются для загрузки делителя скорости обмена. В первый записывается младший, во второй – старший байт делителя. Они формируют шестнадцатибитовый делитель, значение которого вычисляется по следующей формуле:

$$\text{Делитель} = \text{байт}_{\text{регистр 00h}} + 256 \times \text{байт}_{\text{регистр 01h}}.$$

В компьютере тактовая частота, подаваемая в UART, составляет 1,8432 МГц. Внутри UART эталонная частота образуется как тактовая, деленная на 16, и равна 115200 Гц. Соотношение между значениями делителя и скоростью можно представить в виде формулы:

$$\text{Скорость} = \frac{115200}{\text{Делитель}}.$$

Для получения скорости 9600 бод необходимо, чтобы делитель был равен 12. Следовательно, в буферный регистр приемопередатчика (00h) должно быть записано число 12, а в регистр разрешения прерываний (01h) – ноль. Если в регистры делителя записана единица, то получится самая высокая скорость – 115200 бод.

По смещению 04h расположен регистр управления модемом. В общем случае он используется для управления двумя выходами интерфейса – RTS и DTR. Назначения битов регистра следующие:

0 0 0 LOOP OUT2 OUT1 RTS DTR

биты 7–5	всегда нули
LOOP	1 = обратная связь доступна
	0 = обратная связь недоступна

OUT2	1 = включено
	0 = выключено, для внутреннего использования
OUT1	1 = включено
	0 = выключено, для внутреннего использования
RTS	1 = включено
	0 = выключено, присутствует на разъеме RS232
DTR	1 = включено
	0 = выключено, присутствует на разъеме RS232

По смещению 05h находится регистр состояния приемопередатчика, который содержит информацию о состоянии приемника и передатчика UART. При использовании совместно с регистром идентификации прерываний (02h) можно установить источник прерываний. Назначения битов регистра приведены ниже:

```
0 TXE TBE BREK FRME PARE OVFE RxRD
```

TXE (передатчик пуст)	1 = нет байта в регистре передатчика и регистре сдвига
	0 = в регистре передатчика и регистре сдвига один байт
TBE (буфер передатчика пуст)	1 = нет байта в регистре передатчика
	0 = в регистре передатчика один байт
BREK (остановка)	1 = обнаружена остановка
	0 = нет остановки
FRME (ошибка блока)	1 = обнаружена ошибка
	0 = нет ошибки
PARE (ошибка проверки на четность)	1 = обнаружена ошибка
	0 = нет ошибки
OVRE (ошибка переполнения)	1 = обнаружена ошибка
	0 = нет ошибки
RxRD (принятые данные готовы)	1 = принятые данные находятся в регистре приемника
	0 = нет принятых данных

По смещению 06h находится регистр состояния модема, который может использоваться для определения состояния входных сигналов, в частности DCD, DSR, CTS, RI, а также для считывания четырех цифровых входных линий. Назначения битов регистра следующие:

```
DCD RI DSR CTS DDCD DRI DDSR DCST
```

DCD (обнаружена несущая передачи данных)	1 = DCD активна
	0 = DCD неактивна
RI (индикатор звонка)	1 = RI активна
	0 = RI неактивна
DSR (набор данных готов)	1 = DSR активна
	0 = DSR неактивна
CTS (сброс для передачи)	1 = CTS активна
	0 = CTS неактивна

DDCD (дельта DCD)	1 = DCD изменена с момента последнего считывания 0 = DCD без изменений
DRI (дельта RI)	1 = RI изменена с момента последнего считывания 0 = RI без изменений
DDSR (дельта DSR)	1 = DSR изменена с момента последнего считывания 0 = DSR без изменений
DCTS (дельта CTS)	1 = CTS изменена с момента последнего считывания 0 = CTS без изменений

Регистр со смещением 07h – это байт памяти. Запись данных в регистр не влияет на операции UART.

Преобразователи напряжений

Выходные сигналы управления (RTS и DTR) и входные сигналы состояния (CTS, DSR, DCD) последовательного порта инвертированы. Последовательные сигналы данных SIN и SOUT не инвертированы. UART работает только с уровнем напряжений ТТЛ/КМОП. Преобразователи напряжений расположены между UART и разъемом RS232. Преобразователи передатчиков конвертируют уровень напряжения ТТЛ в уровень RS232, а преобразователи приемников – наоборот. Логическая структура последовательного порта изображена на рис. 1.10.

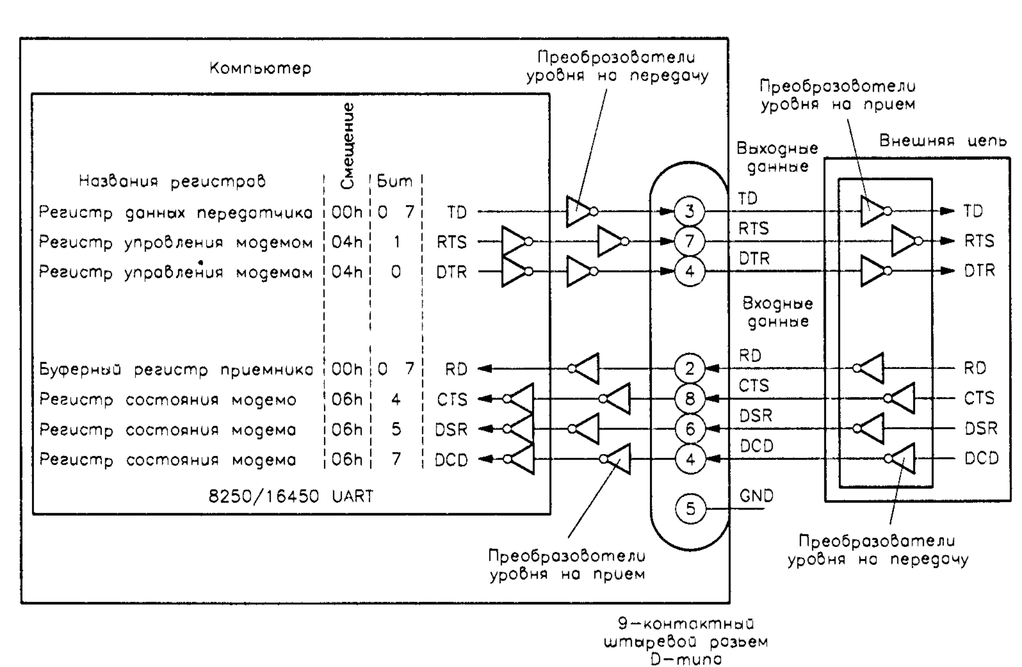


Рис. 1.10. Логическая структура последовотельного порта

Базовые адреса COM-портов

Базовые адреса COM-портов выглядят так:

COM1: 3F8h
COM2: 2F8h
COM3: 3E8h
COM4: 2E8h

При включении или перезагрузке компьютера BIOS проверяет адреса всех установленных последовательных портов. Если она находит такой порт, то заносит базовый адрес (двухбайтовое слово) в определенную ячейку памяти. Для COM1 это ячейки 0000:0400h и 0000:0401h. Базовый адрес можно получить, считав их содержимое. Ячейки памяти, в которых содержится информация о базовых адресах установленных последовательных портов, приведены ниже:

COM1: 0000:0400h – 0000:0401h
COM2: 0000:0402h – 0000:0403h
COM3: 0000:0404h – 0000:0405h
COM4: 0000:0406h – 0000:0407h

Однобайтовая ячейка памяти 0000:0411h в первом, втором и третьем битах содержит общее количество установленных COM-портов:

бит 3 = 0, бит 2 = 0, бит 1 = 0	COM-порты не установлены
бит 3 = 0, бит 2 = 0, бит 1 = 1	установлен один COM-порт
бит 3 = 0, бит 2 = 1, бит 1 = 0	установлено два COM-порта
бит 3 = 0, бит 2 = 1, бит 1 = 1	установлено три COM-порта
бит 3 = 1, бит 2 = 0, бит 1 = 0	установлено четыре COM-порта

1.2.4. Программное управление

В разделе приведены основные варианты программирования последовательного порта.

Получение базового адреса последовательного порта

Представленная программа, написанная на языке QBASIC, выводит общее количество COM-портов, встроенных в ПК, и их базовые адреса. Строка 20 считывает байт из ячейки памяти 0000:0411h, используя команду PEEK(). Биты 0, 1 и 2 содержат информацию о количестве установленных COM-портов. На эти три бита накладывается маска с помощью оператора AND (1+2+4) для определения количества портов. Строка 30 считывает два байта из ячеек памяти, содержащих базовый адрес COM1. Строки 40, 50 и 60 делают то же самое для остальных портов.

```
10 DEF SEG=0
20 PRINT "Number of RS232 ports:",(PEEK(&H411) AND (1+2+4))
30 PRINT "Address of COM1:",PEEK(&H400)+256*PEEK(&H401)
40 PRINT "Address of COM2:",PEEK(&H402)+256*PEEK(&H403)
```

```
50 PRINT "Address of COM3:",PEEK(&H404)+256*PEEK(&H405)
60 PRINT "Address of COM:".PEEK(&H406)+256*PEEK(&H407)
70 INPUT X
```

Следующая функция, написанная на языке TP6, считывает информацию о количестве установленных портов и присваивает полученное значение переменной Number_of_COM. Затем она считывает базовые адреса из ячеек памяти, где они хранятся, и присваивает адрес выбранного порта переменной RS232_address.

```
(*-Библиотека ресурсов № A6 (определение базовых адресов COM-портов).-*)
Procedure COM_address;
(* $0000:$0400 содержит базовый адрес порта COM1,
   $0000:$0402 содержит базовый адрес порта COM2,
   $0000:$0404 содержит базовый адрес порта COM3,
   $0000:$0406 содержит базовый адрес порта COM4,
   $0000:$0411 содержит количество COM-портов в двоичном формате.*)
var
   COM:array[1..4] of integer;
   COM_number,number_of_COM,code:integer;
   Kbchar:char;
begin
   clrscr;
   COM_number:=1;                        (*Установка порта по умолчанию.*)
   Number_of_COM:=mem($0000:$0411);      (*Считывание количества COM-портов.*)
   Number_of_COM:=(Number_of_COM and (8+4+2)) shr 1;
   COM[1]:=memw($0000:$0400);            (*Процедура считывания из памяти.*)
   COM[2]:=memw($0000:$0402);
   COM[3]:=memw($0000:$0404);
   COM[4]:=memw($0000:$0406);
   Textbackground(blue); clrscr;
   Textcolor(yellow); Textbackground(red); window(10,22,70,24); clrscr;
   writeln('Number of COM installed:', Number_of_COM:2);
   writeln('Addresses for COM1 to COM4:', COM[1]:3', COM[2]:3', COM[3]:3', COM[4]:3);
   write('Select COM to be used (1,2.3,4):');
   delay(1000);
   if number_of_COM>1 then  begin (*Выбор конкретного порта, если установлено несколько портов.*)
      repeat
         kbchar:=readkey;                (*Считывание значения с вводимой клавиши.*)
         val(kbchar,COM_number,code);    (*Преобразование символа в число.*)
      until (COM_number>=1) and (COM_number<=4) and (COM[COM_number]<>0);
                             end;
   clrscr;
   RS232_address:=COM[COM_number];
   writeln('Your selected RS232 interface: COM', COM_number:1);
   write('RS232 address:', RS232_address:4);
   delay(1000);
   textbackground(black); window(1,1,80,25); clrscr;
end;
```

В следующем примере приведена функция RS232(X), написанная на языке Turbo Pascal для Windows. RS232(0) возвращает количество установленных COM-портов, RS232(1) – базовый адрес COM1, RS232(2) – базовый адрес COM2 и т.д.

```
(*-Библиотека ресурсов № А6 (определение базовых адресов COM-портов).-*)
(Universal auto detection of COM base address)
Function RS232(X:integer).integer; export;
(* $0000 $0400 содержит базовый адрес порта COM1,
   $0000:$0402 содержит базовый адрес порта COM2,
   $0000:$0404 содержит базовый адрес порта COM3,
   $0000·$0406 содержит базовый адрес порта COM4,
   $0000:$0411 содержит количество COM-портов в двоичном формате.*)
var
   number_of_COM, COM1, COM2,COM3, COM4:integer;
begin
   number_of_COM:=mem($40.$11);     (*Считывает количество COM-портов *)
   number_of_COM:=(number_of_COM and (8+4+2)) shr 1;
   COM1:=0; COM2:=0; COM3:=0; COM4:=0;
   COM1:=memw($40:$00);             (*Процедура чтения из памяти.*)
   COM2.=memw($40:$02);
   COM3:=memw($40:$04),
   COM4:=memw($40:$06);
   Case X of
       0. RS232:=number_of_COM;
       1. RS232:=COM1;
       2: RS232:=COM2;
       3: RS232:=COM3;
       4: RS232:=COM4;
   end;
end,
```

Инициализация COM-порта

Перед тем как использовать COM-порт, его необходимо настроить на определенный формат передачи данных, то есть установить скорость, количество битов данных, количество стоповых битов и бит проверки.

Существует три метода настройки. Первый заключается в использовании команды MODE операционной системы MS DOS. Синтаксис команды можно представить так:

```
MODE COMm: baud=b, parity=p,data=d,stop=s,retry=r
или MODE COMm:b,p,d,s,r
```

MODE COM1:96,n,8,1 настраивает порт COM1 со следующими параметрами: скорость 9600 бод, без проверки на четность, 8 бит данных, 1 стоповый бит. Указанная команда может быть включена в файл autoexec.bat. Недостаток такого метода – невозможность изменять формат передачи данных в пользовательских программах.

Второй метод использует прерывание BIOS INT 14h, которое позволяет выполнять настройку порта из программ пользователя. Для этого необходимо в регистр AH загрузить 0, а в DX – число от 0 до 3, указывающее на соответствующий порт (COM1 – COM4). В регистр AL загружается байт инициализационных данных, значения битов которого приведены ниже:

```
BD2 BD1 BD0 PAR1 PAR0 STOP DA1 DA0
```

BD2 – BD0 (скорость)	111 = 9600
	011 = 600

	110 = 4800
	010 = 300
	101 = 2400
	001 = 150
	100 = 1200
	000 = 110
PAR1,0 (проверка на четность)	00 или 10 = нет проверки
	01 = нечетная
	11 = четная
STOP (количество стоповых битов)	0 = 1
	1 = 2
DA1,0 (длина блока данных)	10 = 7 бит
	11 = 8 бит

Следующая программа на языке TP6 делает то же, что и команда DOS MODE COM1:96,n,8,1.

```
Procedure initialize;
{COM1: 9600, без проверки на четность, 8 бит данных, 1 стоповый бит.}
var
  register:registers;
begin
  with register do begin
                    ah:=0                  {Загрузка номера функции прерывания.}
                    al:=128+64+32+0+0+0+2+1; {Загрузка инициализационного кода 11100011B.}
                    dx:=0;                 {Номер порта: DX=0:COM1, DX=1:COM2 и т.д.}
                    intr($14,register);    {Вызов прерывания BIOS.}
                 end;
end;
```

Ограничение описанного метода состоит в том, что можно задать скорость только 9600 бод. UART 16450 способен работать со скоростью 115200 бод, это достигается непосредственным обращением к регистру.

Третий, наиболее гибкий метод настраивает порт посредством записи данных в регистр формата данных UART (смещение 03h). Следующая программа на TP6 позволяет настроить сам регистр, для этого требуется базовый адрес настраиваемого порта, скорость, режим проверки, длина блока данных и количество стоповых битов. Процедура переводит заданную скорость в шестнадцатибитовый делитель и загружает его в соответствующие регистры.

```
(*-Библиотека ресурсов № A9 (запись в регистр формата данных).-*)
Procedure Write_data_format (RS232_address, Baud, Parity, Data_bit, Stop_bit:integer);
var
  byte1,byte2,output_byte:byte;
  divisor:integer;

begin
  divisor:=115200 div Baud;
```

```
    if divisor<=255 then begin byte1:=divisor; byte2:=0; end;
    if divisor>255 then begin byte1:=divisor mod 256; byte2:=divisor div 256; end;
    output_byte:=(data_bit-5)+4*(stop_bit-1)+8*(parity);
    port(RS232_address+3):=128;            {Загрузка инициализационных данных, первый бит
                                            регистра равен 1.}
    port(RS232_address+0):=byte1;          {Младший байт делителя равен 1.}
    port(RS232_address+1):=byte2;          {Старший байт делителя равен 0.}
    port(RS232_address+3):=output_byte;    {Загрузка делителя и других параметров.}
end;
```

Следующая функция, написанная на языке Turbo Pascal для Windows, выполняет то же самое.

```
(*-Библиотека ресурсов № A9 (запись в регистр формата данных).-*)
Function Write_data_format(RS232_address, Baud, Parity, Data_bit, Stop_bit:integer):integer; export;
var
    byte1,byte2,output_byte:byte;
    divisor:integer;
begin
    divisor:=115200 div Baud;
    if divisor<=255 then begin byte1:=divisor; byte2:=0; end;
    if divisor>255 then
    begin
        byte1:=divisor mod 256;
        byte2:=divisor div 256;
    end;
    output_byte:=(data_bit-5)+4*(stop_bit-1)+8*parity;
    port(RS232_address+3):=128;            {Загрузка инициализационных данных,
                                            первый бит регистра равен 1.}
    port(RS232_address+0):=byte1;          {Младший байт делителя равен 1.}
    port(RS232_address+1):=byte2;          {Старший байт делителя равен 0.}
    port(RS232_address+3):=output_byte;    {Загрузка делителя и других параметров.}
end;
```

Передача и прием последовательных данных

Существует несколько способов приема и передачи данных через последовательный порт: с помощью команд операционной системы, прерываний BIOS или непосредственного доступа к порту. Последний способ наиболее удобен при проведении операций ввода/вывода общего назначения. Рассмотрим пример для порта COM1. Чтобы передать данные, можно записать их непосредственно в буферный регистр передатчика 3F8h, используя следующий оператор языка QBASIC:

```
OUT 3F8h, X
```

где X – данные в десятичном формате. Для получения данных из порта COM1 считываются данные из буферного регистра приемника 3F8h. С этой целью используется другой оператор языка QBASIC (Y – входные данные в десятичном формате):

```
Y=INP(3F8h)
```

Следующие две процедуры написаны на языке TP6 и выполняют те же функции.

```
(*Библиотека ресурсов № A10 (запись данных в буферный регистр передатчика).*)
Procedure write_transmit_buffer(RS232_address,output_byte:integer),
begin
  port(RS232_address):=output_byte;
end;

(*Библиотека ресурсов № A12 (чтение данных из буферного регистра приемника) *)
Function read_receive_buffer(RS232_address,output_byte:integer):integer;
begin
  read_receive_buffer:=port(RS232_address);
end;
```

Ниже приведены две функции, написанные на Turbo Pascal для Windows.

```
(*Библиотека ресурсов № A10 (запись данных в буферный регистр передатчика).*)
Function write_transmit_buffer(RS232_address,output_byte integer).integer; export;
begin
  port(RS232_address):=output_byte;
end;

(*Библиотека ресурсов № A11 (чтение данных из буферного регистра приемника).*)
Function read_reseive_buffer(RS232_address,output_byte:integer):integer, export;
begin
  read_reseive_buffer:=port(RS232_address);
end;
```

Передача данных по линиям взаимодействия

Для вывода данных через линии RTS и DTR в регистр управления модемом (смещение 04h) необходимо записывать биты 1 и 0, которые соответствуют сигналам RTS и DTR. Линии управляются процедурами на языках TP6 и Turbo Pascal для Windows, требующими базовый адрес выбранного COM-порта и состояние этих линий – либо 0, либо 1. Причем RTS и DTR инвертируются перед подачей в порт с целью компенсации инверсии преобразователями ТТЛ/RS232, которые также используются для трансформации уровня напряжения.

```
(*-Библиотека ресурсов № A11 (запись данных в регистр состояния модема) -*)
procedure write_modem_status(RS232_address, RTS. DTR:integer);
(*RTS и DTR инвертируются с помощью MAX238 на экспериментальной плате *)
(*RTS=бит 1, DTR=бит 0 регистра управления модемом, смещение 04h.*)
begin
  RTS.=1-RTS,
  DTR:=1-DTR;
  port(RS232_address+4):=RTS*2+DTR;          (*Запись в регистр 04h *)
end;

(*-Библиотека ресурсов № A11 (запись данных в регистр состояния модема).-*)
Function write_modem_status(RS232_address, RTS, DTR:integer):integer; export;
(*RTS и DTR инвертируются с помощью MAX238 на экспериментальной плате.*)
```

```
(*RTS=бит 1, DTR=бит 0 регистра управления модемом, смещение 04h.*)
begin
  RTS:=1-RTS;
  DTR:=1-DTR;
  port(RS232_address+4):=RTS*2+DTR;    (*Запись в регистр 04h *)
end,
```

Чтобы считать данные с линий DSR, CTS и DCD, необходимо считать регистр состояния модема. Для этого служат нижеприведенные процедуры на языках TP6 и TPW, требующие базовый адрес выбранного COM-порта. Линии DSR, CTS и DCD инвертируются для компенсации инверсии преобразователями ТТЛ/RS232.

```
(*-Библиотека ресурсов № A13 (чтение данных из регистра состояния модема).-*)
Function read_modem_status(RS232_address, x:integer):integer;
(*x=1 - выбор бита DCD, x=2 - выбор бита DSR, x=3 - выбор бита CTS.*)
(*DCD=бит 7, DSR=бит 5, CTS=бит 4 регистра состояния модема, смещение 06h *)
(*Все биты инвертируются с помощью MAX238 на экспериментальной плате.*)
var
  input_byte byte;
begin
  input_byte:=port(RS232_address+6),
  case x of
     1:  Read_modem_status =1-round((input_byte and 128)/128);
     2:  Read_modem_status.=1-round((input_byte and 32)/32);
     3:  Read_modem_status =1-round((input_byte and 16)/16),
  end;
end;

(*-Библиотека ресурсов № A13 (чтение данных из регистра состояния модема) -*)
Function read_modem_status(RS232_address, x:integer):integer; export;
(*x=1 - выбор бита DCD, x=2 - выбор бита DSR, x=3 - выбор бита CTS.*)
(*DCD=бит 7, DSR=бит 5, CTS=бит 4 регистра состояния модема, смещение 06h.*)
(*Все биты инвертируются с помощью MAX238 на экспериментальной плате.*)
var
  input_byte·byte;
begin
  input_byte:=port(RS232_address+6);
  case x of
     1:  Read_modem_status:=1-round((input_byte and 128)/128);
     2:  Read_modem_status:=1-round((input_byte and 32)/32);
     3:  Read_modem_status:=1-round((input_byte and 16)/16);
  end;
end;
```

1.3. Игровой порт

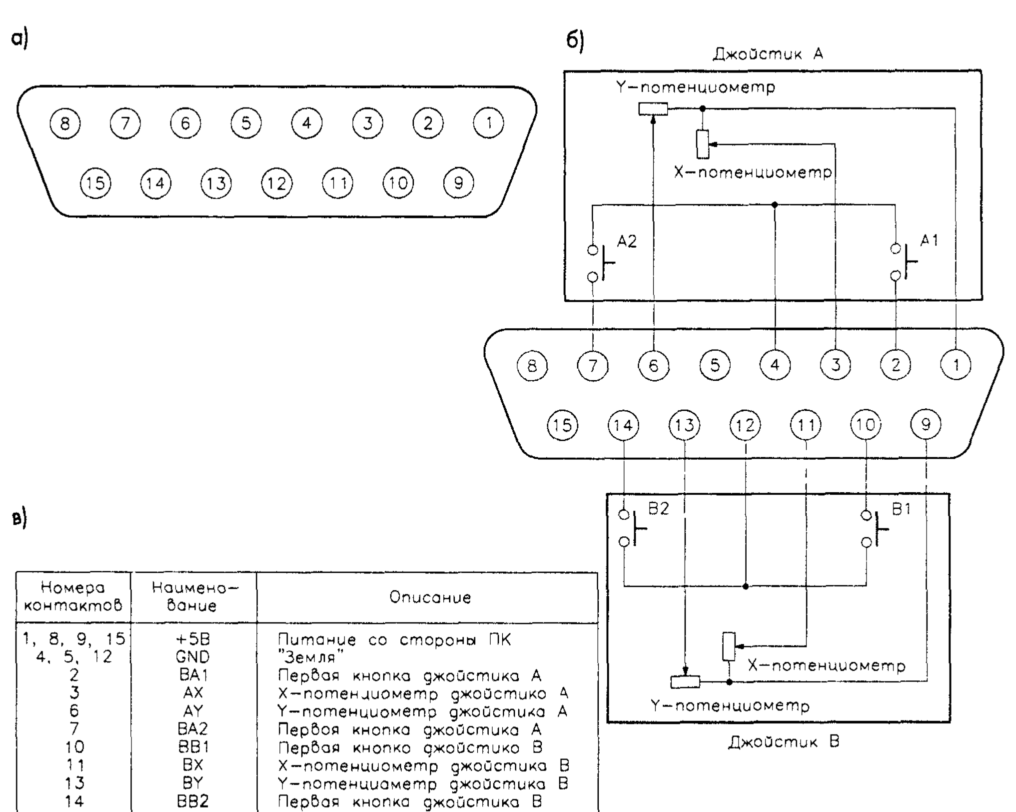
В большинстве настольных компьютеров есть встроенный игровой порт, куда подключаются один или два джойстика – с двумя цифровыми и двумя аналоговыми входными линиями (одно устройство) или с четырьмя цифровыми и четырьмя аналоговыми (два устройства). Цифровые линии позволяют считывать

входные данные, а аналоговые – измерять подключенное к ним сопротивление (оно должно быть в пределах 0–100 кОм).

Джойстик имеет два потенциометра 100 кОм для индикации координат X и Y, а также две нормально разомкнутые кнопки. На соответствующих им линиях порта устанавливается высокий уровень, а при нажатии на кнопку – низкий.

1.3.1. Разъем

Порт имеет 15-контактную розетку D-типа. Назначение выводов и типовое соединение с джойстиком представлены на рис. 1.11.



Номера контактов	Наименование	Описание
1, 8, 9, 15	+5В	Питание со стороны ПК
4, 5, 12	GND	"Земля"
2	BA1	Первая кнопка джойстика А
3	AX	X-потенциометр джойстика А
6	AY	Y-потенциометр джойстика А
7	BA2	Первая кнопка джойстика А
10	BB1	Первая кнопка джойстика В
11	BX	X-потенциометр джойстика В
13	BY	Y-потенциометр джойстика В
14	BB2	Первая кнопка джойстика В

Рис. 1.11. Назначение выводов и типовое соединение с джойстиком: а – 15-контактный гнездовой разъем D-типа, вид со стороны задней стенки компьютера, б – соединение контактов порта с внешними резисторами и переключателями; в – назначение контактов игрового порта

1.3.2. Внутреннее аппаратное устройство

Внутренняя блок-схема типового игрового порта изображена на рис. 1.12, а логическая структура – на рис. 1.13.

Восьмибитовые линии шины данных состоят из четырех выходов микросхемы DD5a и четырех выходов микросхемы DD5b. Четыре входа состояния кнопок

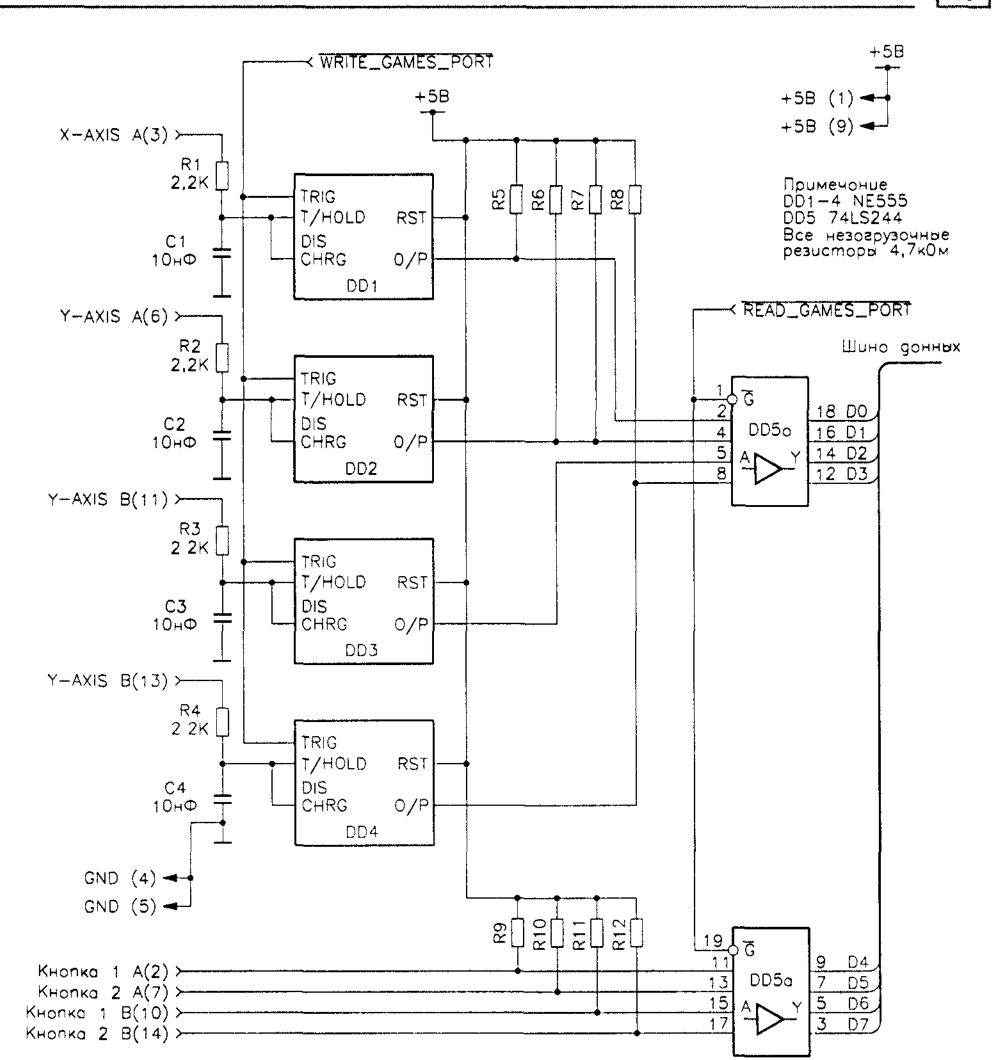


Рис. 1.12. Внутренняя блок-схема типового игрового порта

подключены к питанию +5 В через нагрузочные резисторы. Порт имеет адрес 201h. Функции битов порта приведены ниже:

```
BB2 BB1 BA2 BA1 BY BX AY AX
```

BB2, BB1, BA2 и BA1	цифровые входы
BY, BX, AY и AX	аналоговые выходы

Метод измерения сопротивления основан на программном определении длительности импульса, пропорционального сопротивлению. Преобразование начинается после вывода любого байта в регистр порта (201h), при этом биты 0–3 устанавливаются в 1. Время измеряется до возврата в нулевое состояние битов 0–3,

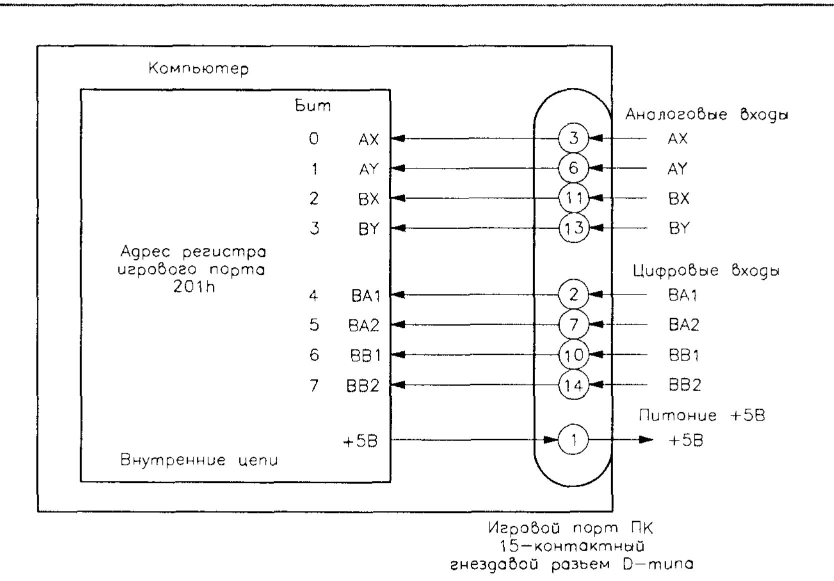


Рис. 1.13. Логическая структура игрового порта

соотнесенных с четырьмя аналоговыми каналами. Если аналоговый вход закорочен на «землю» или цепь измеряемого сопротивления разорвана, соответствующий бит не обнулится. Поэтому в программе преобразования должен быть предусмотрен тайм-аут. Взаимосвязь временного интервала и подключенного сопротивления характеризуется следующей формулой:

$$\text{Сопротивление (Ом)} = \frac{\text{Временной интервал} - 24{,}2\ \text{мкс}}{0{,}011}.$$

Временной интервал изменяется в пределах от 24,2 мкс для нулевого сопротивления до 1124 мкс для максимального (100 кОм). Однако погрешности внутреннего конденсатора и резистора делают формулу неточной. На практике необходимо производить калибровку, которая включает в себя измерение интервалов времени, когда входное сопротивление близко к нулю или равно 100 кОм.

1.3.3. Программное управление

В языке программирования QBASIC существуют два оператора, предназначенные для игрового порта, – `STICK(x)` и `STRIG(x)`. Оператор `STICK(x)` может принимать значения 0–3 и используется для чтения информации с потенциометров X и Y джойстиков A и B.

`X=0` считывает X-координату джойстика A
`X=1` считывает Y-координату джойстика A
`X=2` считывает X-координату джойстика B
`X=3` считывает Y-координату джойстика B

Применяя эту команду, необходимо сначала вызвать `STICK(0)`, а затем `STICK(1)`, `STICK(2)` и `STICK(3)`. Функция `STICK` возвращает значение координаты, которое изменяется от 6 (нулевое сопротивление) до 150 (сопротивление 100 кОм).

STRIG(x) возвращает −1, если условие верно; в противном случае функция возвращает 0. X может принимать значения от 0 до 7 и используется для выбора определенного состояния кнопки джойстика.

X=0 Первая кнопка джойстика A была нажата с момента последней команды STRIG(0)
X=1 Первая кнопка джойстика A была только что нажата
X=2 Первая кнопка джойстика B была нажата с момента последней команды STRIG(2)
X=3 Первая кнопка джойстика B была только что нажата
X=4 Вторая кнопка джойстика A была нажата с момента последней команды STRIG(4)
X=5 Вторая кнопка джойстика B была только что нажата
X=6 Вторая кнопка джойстика A была нажата с момента последней команды STRIG(6)
X=7 Вторая кнопка джойстика B была только что нажата

Следующая программа на языке QBASIC выводит на экран координаты X и Y джойстика A и показывает состояния двух кнопок.

```
10 dummy=STICK(0)
20 print Coordinate of X  ,STICK(0)
30 print Coordinate of Y  ,STICK(1)
40 print Current status of 1st button   STRIG(1)
50 print Current status of 2nd button  ,STRIG(5)
60 end
```

Функция на языке TP6, приведенная ниже, возвращает состояние конкретного бита, определенного переменной Bitx. Содержимое регистра игрового порта считывается в компьютер и присваивается переменной Input_byte. Затем состояние конкретного бита определяется наложением маски на считанный байт.

```
(*-Библиотека ресурсов № A14 (чтение регистра игрового порта) -*)
Function read_game_port(bitx integer) integer,
(*Адрес игрового порта  201h
Bitx (1-8) выбирает состояние AX, AY, BX, BY, BA1, BA2, BB1 и BB2 *)
var
   input_byte byte
begin
   input_byte =port($201),
   read_game_port =round((input_byte and bit_weight(bitx))/bit_weight(bitx)),
end,
```

Чтобы определить величину сопротивления, необходимо записать байт в порт 201h для запуска одновибратора. Соответствующий бит в регистре порта становится единицей. Нужно постоянно опрашивать этот бит, пока он снова не станет нулем. В результате получим требуемый временной интервал. Наиболее удобно измерять временной интервал с помощью третьего счетчика/таймера 8253/8254 внутри компьютера, который можно настроить как независимый таймер с обратным отсчетом времени. Если в счетчике записано число 255, его значение обнуляется

каждые 55 мс. Разрешается применять только третий счетчик, так как первые два уже используются операционной системой компьютера.

Следующая функция на языке TP6 позволяет измерить временной интервал. Переменная x указывает, по какому аналоговому входу он будет определяться. Функция port($201):=0 выполняет операцию записи в игровой порт для запуска одновибратора. Сразу после этого считывается значение счетчика 8253 и полученный результат присваивается переменной Time1. Потом в цикле постоянно проверяется, когда соответствующий бит регистра станет равным нулю. Как только это произойдет, значение счетчика снова считывается и присваивается переменной Time2. Затем вычисляется временной интервал. Данная функция использует процедуру init_8253 и функцию Read_8253. Первая записывает число 255 в старший и младший разряды счетчика 3 и настраивает его как независимый счетчик, вторая считывает старший и младший байты счетчика 3.

```
(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
  time1,time2,dummy:integer;

Procedure init_8253;
(*Настройка 8253.*)
begin
(* Управляющее слово = b6h = 10110110b:
  10 = выбор счетчика 2;
  11 = сначала чтение/запись младшего байта, затем старшего;
  011 = режим 3;
  0 = шестнадцатибитовый двоичный счет.*)
  port($43):=$b6;             (*Запись управляющего слова в управляющий регистр 8253.*)
  port($42):=255;             (*Загрузка младшего байта.*)
  port($43):=255:             (*Загрузка старшего байта.*)
  port($61):=port($61) or 1;  (*Выключение внутреннего динамика.*)
  port($43):=$80:             (*80h - команда фиксации для счетчика 3.*)
end;

Function read_8253:integer;

(*Считывание младшего и старшего байтов счетчика.*)
var
  low_byte,high_byte:byte;
begin
  low_byte:=port($42);
  high_byte:=port($43);
  read_8253:=low_byte+256*high_byte;
end;

var
  i:integer;
begin
  init_8253;
  for i:=1 to 100 do i:=i;
```

```
  i:=0;
  dummy:=bit_weight(x);
  port($201):=0;
  time1:=read_8253;
  repeat i:=i+1 until (port($201) and dummy=0) or (i>=5000);
  time2:=read_8253;
  interval_game_port:=time2-time1;
  if i>=5000 then interval_game_port:=0;
end;
```

А вот как выглядит эта функция, написанная на языке Turbo Pascal для Windows:

```
(*-Библиотека ресурсов № A14 (чтение регистра игрового порта).-*)
Function read_game_port(bitx:integer):integer: export;
(*Адрес игрового порта: 201h
Bitx выбирает состояние AX, AY, BX, BY, BA1, BA2, BB1 и BB2.*)
var
  input_byte byte;
begin
  input_byte:=port($201);
  read_game_port:=round((input_byte and bit_weight(bitx))/bit_weight(bitx));
end;

(*-Библиотека ресурсов № A15 (запись в регистр игрового порта).-*)
Function write_game_port:integer; export;
(*Вводит 0 в игровой порт для запуска мультивибратора.*)
begin
  port($201):=0;
end,

(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer; export;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
  time1,time2.dummy:integer,

Procedure init_8253,
(*Настройка 8253.*)
begin
(* Управляющее слово = b6h = 10110110b
  10 = выбор счетчика 2;
  11 = сначала чтение/запись младшего байта, затем старшего;
  011 = режим 3;
  0 = шестнадцатибитовый двоичный счет.*)
  port($43):=$b6;              (*Запись управляющего слова в управляющий регистр 8253.*)
  port($42).=255;              (*Загрузка младшего байта.*)
  port($43).=255;              (*Загрузка старшего байта.*)
  port($61).=port($61) or 1;   (*Выключение динамика.*)
  port($43):=$80;              (*80h - команда фиксации для счетчика 3.*)
end;

Function read_8253:integer;
 (*Считывание младшего и старшего байтов счетчика.*)
var
  low_byte,high_byte:byte;
```

```
begin
  low_byte =port($42),
  high_byte =port($43),
  read_8253 =low_byte+256*high_byte,
end,

var
  i integer,
begin
  init_8253,
  for i =1 to 10 do i =1,
  i =0,
  dummy =bit_weight(x),
  port($201) =0,
  time1 =read_8253,
  repeat i =i+1 until (port($201) and dummy=0) or (i>=10000),
  time2 =read_8253,
  interval_game_port =time2-time1
  if i>=10000 then interval_game_port =0,
end,
```

2. НЕОБХОДИМОЕ ОБОРУДОВАНИЕ

Цифровые интегральные схемы и операционные усилители требуют напряжения питания +5, +12, –5 и –12 В. Кроме того, для измерения напряжения, тока, сопротивления и других физических характеристик, а также для наблюдения формы сигнала необходимы различные приборы: мультиметры, осциллографы и логические пробники. В частности, в опытах с цифровыми устройствами при определении логического состояния линий и детектирования импульсов применяются логические пробники (см. раздел 2.2); аналоговые и цифровые генераторы используются для получения некоторых видов сигналов (см. раздел 2.3). Все опыты по сопряжению, описанные в этой книге, проводятся с помощью трех экспериментальных плат: для параллельного, последовательного и игрового портов. Их конструкция представлена в разделе 2.4.

2.1. Источники питания

В разделе приведены основные способы получения стабилизированного напряжения положительной и отрицательной полярности для электропитания устройств, описанных в книге.

2.1.1. Источник питания постоянного тока

Для работы экспериментальных плат необходим любой источник питания постоянного тока, в том числе батареи, на 8–15 В, 1 А.

На рис. 2.1 изображена схема двуполярного источника питания, формирующего постоянное напряжение +16 и –16 В с номинальным током 1,8 А. Силовая часть источника питания состоит из выключателя, переключателя напряжения 110/220 В, предохранителя и трансформатора мощностью 50 Вт с двумя независимыми первичными обмотками, которые можно соединить последовательно (для подключения к сети переменного тока 220 В) или параллельно (для подключения к сети на 110 В). Стандартный плавкий предохранитель на 3 А включается последовательно с первичной обмоткой. Во вторичной обмотке используются восстанавливаемые

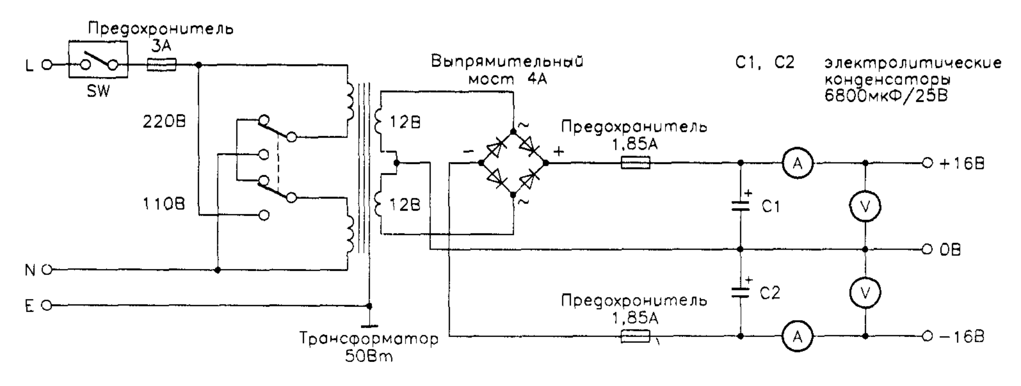


Рис. 2.1. Источник питания +16 и −16 В

предохранители типа RS183-9629: как только величина тока превышает граничное значение, их сопротивление становится очень высоким, а когда ток падает, они автоматически возвращаются в исходное состояние.

2.1.2. Источники питания +5, −5, +12, −12 В

Получить постоянное напряжение проще всего при помощи *стабилитронов*. Например, стабилитроны серии BZX79 позволяют регулировать напряжение в пределах 2,4–75 В при номинальной мощности 500 мВт и точности стабилизации 5%.

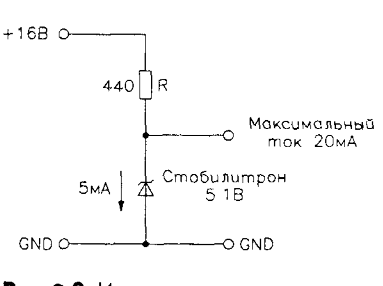


Рис. 2.2. Источник питания с напряжением +5 В, использующий стабипитрон

На рис. 2.2 изображена схема, которая преобразовывает постоянное напряжение 16 В в напряжение 5,1 В с номинальным током 20 мА.

Для получения постоянного напряжения в основном применяют *стабилизаторы напряжения* серий 78 (положительное напряжение) и 79 (отрицательное напряжение). Они могут стабилизировать различные напряжения (+5, +9, +12, +15, +24, −5, −9, −12, −15, −24 В и т.д.) с точностью 5%. Номинальный ток для серий 78L и 79L составляет 100 мА, для 78 и 79 – 1 А, а для 78S и 79S – 2 А. Все стабилизаторы имеют автоматическую тепловую защиту от перегрева. На рис. 2.3 показана схема, формирующая напряжения +5, +12, −5 и −12 В.

В качестве источника входного напряжения можно использовать схему, изображенную на рис. 2.1. В схеме на рис. 2.3 применяются стабилизаторы серий 78 и 79. Все стабилизаторы необходимо установить на радиаторы, а на каждом выходе включить восстанавливаемые предохранители на 1 А.

Поскольку на стабилизаторах большое падение напряжения, следовательно, входное напряжение должно быть на 2–3 В больше, чем выходное. Эти устройства в холостом режиме работы потребляют относительно большой ток, обычно 1–8 мА. Существуют другие типы стабилизаторов, имеющие малое падение напряжения.

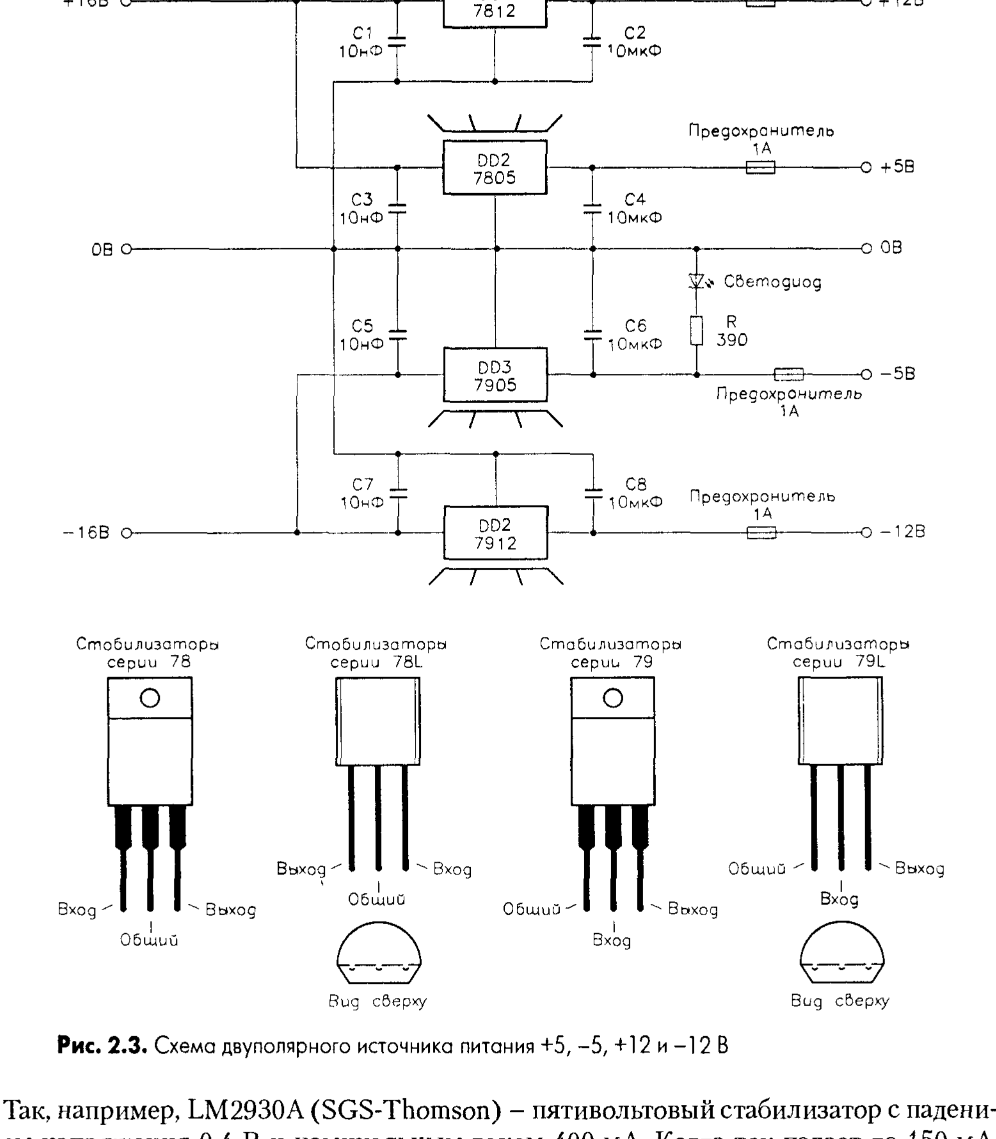


Рис. 2.3. Схема двуполярного источника питания +5, –5, +12 и –12 В

Так, например, LM2930A (SGS-Thomson) – пятивольтовый стабилизатор с падением напряжения 0,4 В и номинальным током 400 мА. Когда ток падает до 150 мА, падение напряжения уменьшается до 0,2 В. Этот стабилизатор также снабжен дополнительными средствами защиты (от превышения входного напряжения 40 В, изменения полярности, а также термозащитой и ограничителем тока). В холостом режиме потребляемый ток равен 22 мА. LM2940CT (National Semiconductor) – стабилизатор с малым падением напряжения, который имеет номинальное напряжение

+5 В и ток 1 А и потребляет ток порядка 3 мА при разнице входного и выходного напряжения 3 В. Если разница напряжений становится менее 3 В, статический ток возрастает до 10 мА. Падение напряжения изменяется от 0,5 до 1 В. На рис. 2.4 показано расположение выводов этих стабилизаторов и представлены типовые схемы включения.

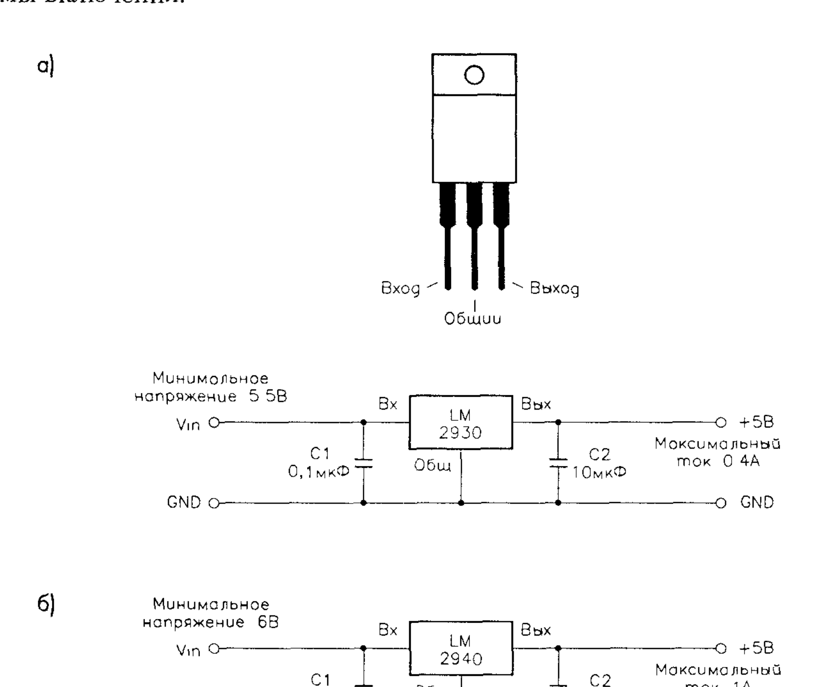
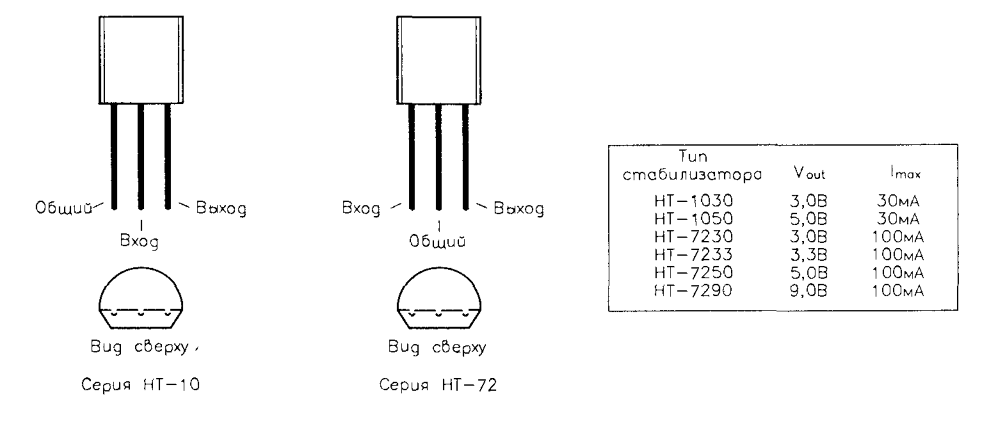


Рис. 2.4. Стабилизаторы LM2930A и LM2940CT и типовые схемы включения а – LM2930A, б – LM2940CT

Стабилизаторы HT-7230, HT-7233, HT-7250 и HT-7290 (Holtek) дают фиксированные значения напряжений 3,0; 3,3; 5,0 и 9,0 В с точностью 5%. Максимальный выходной ток равен 100 мА. Падение напряжения составляет 100 мВ, а ток холостого хода – 500 мкА. Стабилизаторы HT-1030 и HT-1050 имеют ток холостого хода 3,5 мкА и формируют напряжение 3 и 5 В. Номинальный ток равен 30 мА. На рис. 2.5 приведено расположение выводов и типовая схема их включения.

Широко известны также *регулируемые стабилизаторы напряжения*, например L200C (SGS-Thomson) – см. рис. 2.6. При использовании их необходимо установить на радиатор.



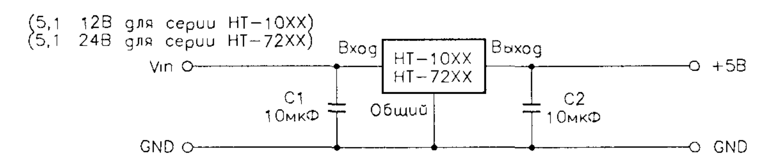


Рис. 2.5. Стабилизаторы напряжения серий HT-10XX и HT-72XX

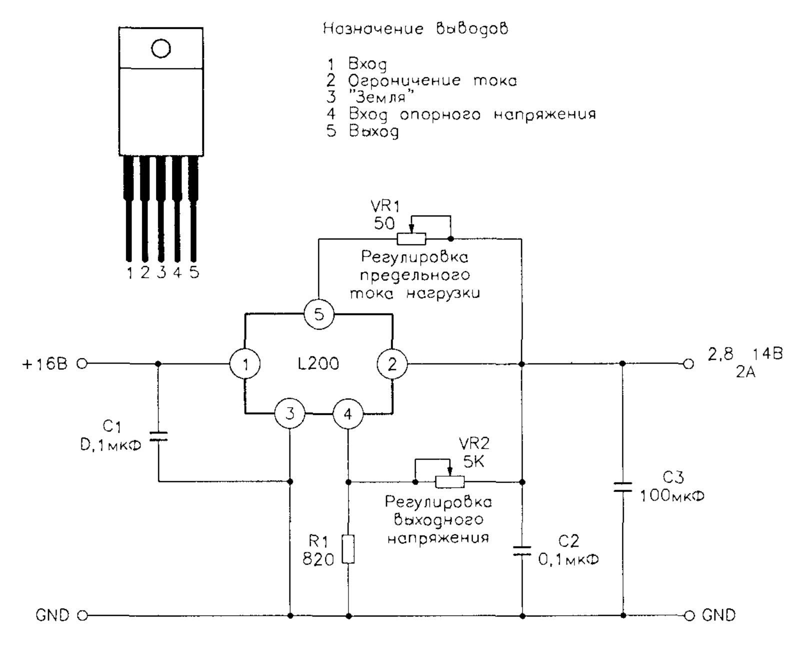


Рис. 2.6. Регулируемый стабилизатор напряжения L200C и его типовое включение

На выходе такой стабилизатор формирует регулируемое напряжение в пределах 2,85–36 В при выходном токе до 2 А. Он имеет защиту по току, термозащиту и защиту от превышения входного напряжения 60 В. Ток холостого хода равен 4,2 мА.

2.1.3. Опорные напряжения

В аналого-цифровых и цифро-аналоговых преобразователях необходимы точные *опорные напряжения*. REF-03CNB, REF-02CP и REF-01CP (Analog Devices) дают напряжения 2,5, 5,0 и 10,0 В с точностью 1%. Максимальный выходной ток равен 21 мА, ток холостого хода – 1 мА; входное напряжение должно быть как минимум на 2 В больше выходного. На рис. 2.7 представлено расположение выводов этих микросхем (все они имеют защиту от короткого замыкания).

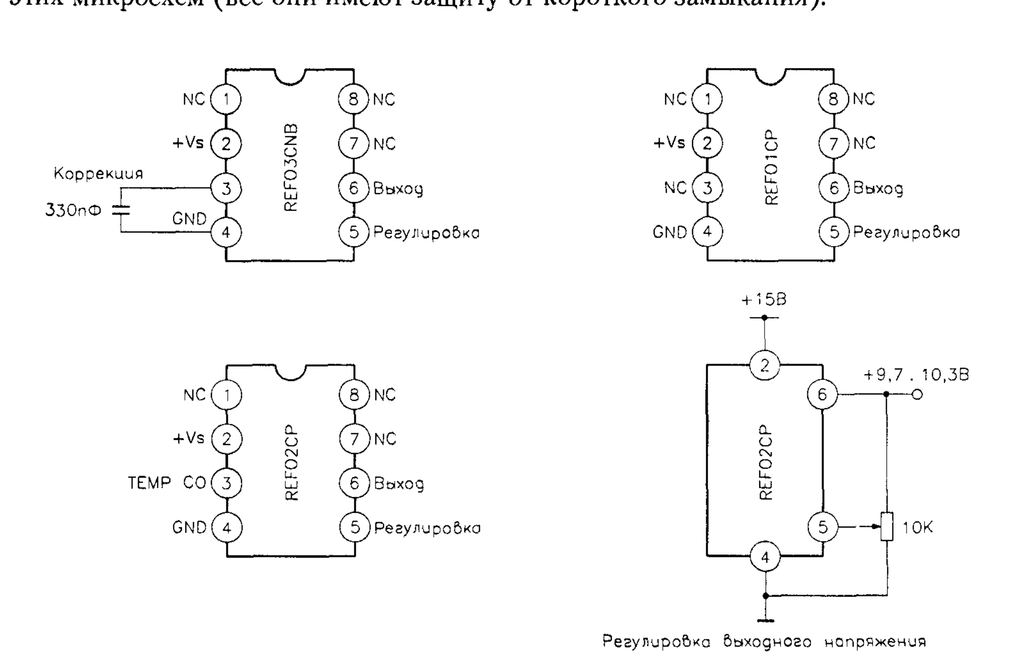


Рис. 2.7. Микросхемы источников опорного напряжения серий REF-XX

Серии LM4040-XX (National Semiconductor) – это маломощные *источники опорного напряжения*, дающие несколько значений напряжения: 2,500; 4,096; 5,000; 8,192 и 10,000 В – и имеющие следующие степени точности: A – 0,1%, B – 0,2%, C – 0,5%, D – 1% и E – 2%. Ток потребления этих устройств изменяется от 60 мкА для напряжения 2,5 В до 100 мкА для 10 В. Все варианты исполнения рассчитаны на максимальный ток 15 мА. Расположение выводов и типовая схема включения приведены на рис. 2.8.

Микросхема TLE2425CLP (Texas Instruments) – другой источник опорного напряжения, формирующий напряжение 2,5 В с точностью 0,8%. Максимальный ток 20 мА, ток холостого хода 170 мкА. Входное напряжение изменяется в пределах 4–40 В. Расположение выводов и типовая схема включения показаны на рис. 2.9.

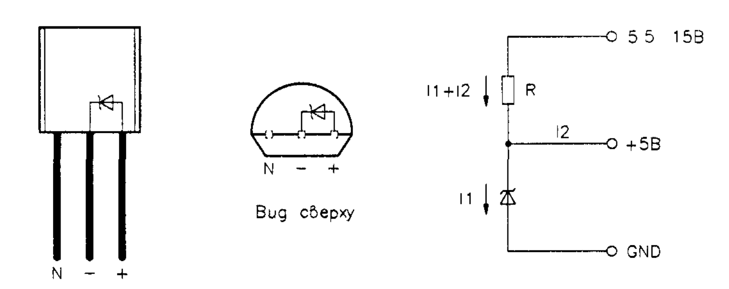


Рис. 2.8. Выводы и типовое включение источников опорного напряжения серии LM4040

Регулируемый генератор опорного напряжения изображен на рис. 2.10. Он строится на основе переменного резистора и широко используется при тестировании АЦП и ЦАП. Выходное напряжение изменяется от нескольких милливольт до нескольких вольт. Для контроля напряжения необходимы высокоточные цифровые вольтметры.

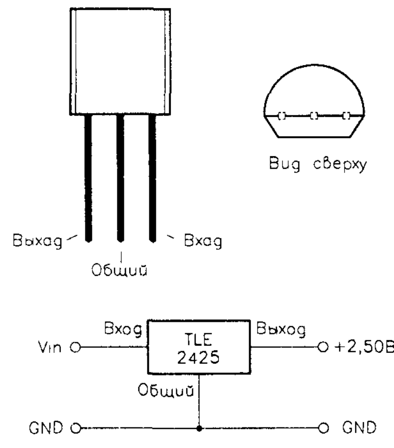


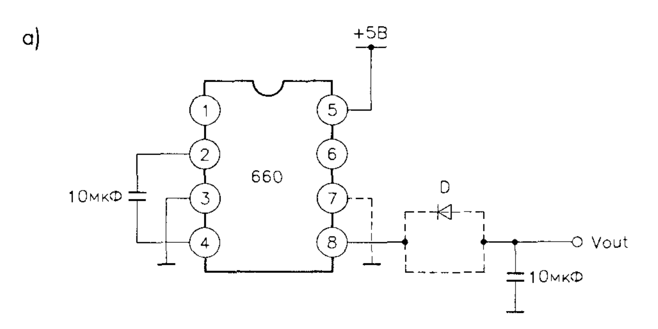
Рис. 2.9. Расположение выводов и типовое включение TLE2425

2.1.4. Преобразователи напряжения

На рис. 2.11а представлена схема *инвертора напряжения*, который трансформирует +5 В в –5 В посредством преобразователя SI660CJ (Siliconix). Микросхема способна генерировать отрицательное напряжение, равное по модулю входному положительному в диапазоне 1,5–10 В.

Для напряжения менее 3,5 В вывод 7 необходимо соединить с «землей», а для напряжения более 6,5 В – подключить диод последовательно с выходом. Выход имеет внутреннее сопротивление 70 Ом. При выходном токе 10 мА напряжение составит 4,3 В. Потребляемый ток в режиме холостого хода равен 170 мкА, а максимальный выходной ток – 40 мА.

Рис. 2.10. Регулируемый генератор напряжения



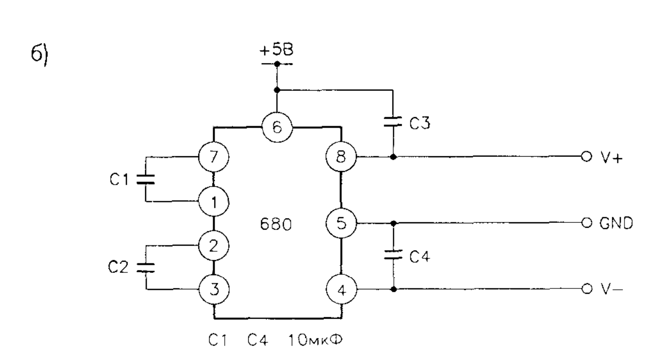


Рис. 2.11. Схемы преобразователей напряжения а – инвертор напряжения 660, б – удвоитель и инвертор напряжения MAX680

Схема, изображенная на рис. 2.11б, преобразует напряжение +5 В в напряжение +10 и –10 В, используя *удвоитель* и инвертор напряжения MAX680CPA (Maxim). Входное напряжение должно быть в пределах 2–6 В. Внутреннее сопротивление для положительного и отрицательного сопротивления равно 150 и 90 Ом соответственно. Если на обоих выходах ток достигнет 10 мА, то напряжение на положительном выходе упадет до 7 В, а отрицательное станет равным –6,1 В. Потребляемый ток в режиме холостого хода составляет 1 мА.

2.1.5. Схемы источников питания с гальванической развязкой

Эти схемы используются в случаях, когда необходима полная изоляция двух цепей[1]. NME и NMA (Newport Components) – высокоэффективные преобразователи напряжения, выходы и входы которых изолированы друг от друга. Разница потенциалов между входом и выходом может достигать 1000 В и более. Преобразователи серии NME работают с напряжением 5 или 12 В и обеспечивают изоляцию и выходное напряжение 5, 12 или 15 В в зависимости от типа. Пятивольтовые

[1] Следует отметить, что такая развязка, как правило, осуществляется с использованием малогабаритных трансформаторов, которые помещаются внутрь корпуса микросхемы. – *Прим. науч. ред.*

преобразователи имеют выходной ток до 200 мА, двенадцативольтовые – 84 мА и пятнадцативольтовые – 67 мА. Преобразователи серии NMA формируют на выходе двуполярное напряжение ±5, ±12 и ±15 В при входном напряжении 5 и 12 В. Максимальный выходной ток пятивольтовых преобразователей равен 100 мА, а пятнадцативольтовых – 42 мА. Назначение выводов этих устройств показано на рис. 2.12.

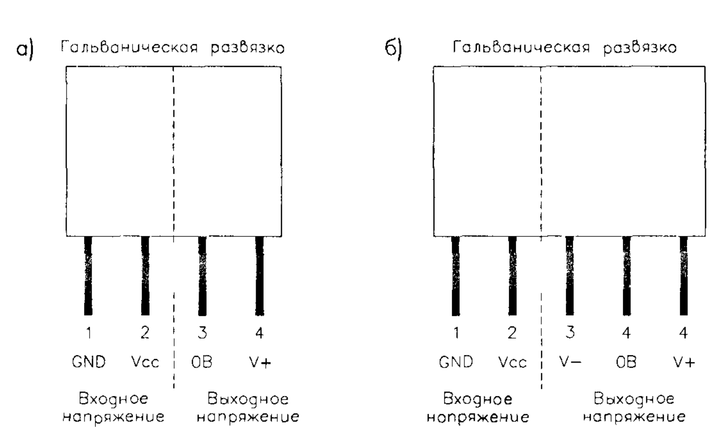


Рис. 2.12. Преобразователи напряжения с гальванической развязкой: а – серия NME, однополярные; б – серия NMA, двуполярные

2.2. Логические пробники

Простой логический пробник можно построить на базе таких буферных микросхем, как 74LS244 или 74LS245 (рис. 2.13а). Ограничительные резисторы включены последовательно со светодиодами. Когда на входе буфера высокий уровень, на выходе он тоже высокий; при этом загорается соответствующий светодиод. Описываемый пробник обеспечивает тесты одновременно по нескольким каналам, однако он не способен фиксировать состояние переходных процессов и высокочастотные импульсные последовательности.

Схема усовершенствованного логического пробника показана на рис. 2.13б. Он строится на базе компаратора напряжения LM339 и имеет три светодиода: красный, желтый и зеленый. При обнаружении высокого уровня загорается красный светодиод, низкого – зеленый, а во время переходных процессов – желтый. Если на вход поступает последовательность импульсов, одновременно горят красный и зеленый светодиоды. Однако и такая схема не способна регистрировать очень короткие импульсы. Для этого необходимо использовать одновибраторы.

2.3. Цифровые и аналоговые генераторы сигналов

В данном разделе приводятся основные сведения по цифровым и аналоговым генераторам сигналов, которые используются для тестирования устройств, представленных в книге.

а)

+5В

Для стандартных светодиодов R = 200 Ом
Для слаботочных светодиодов R = 2кОм

Цифровые входы

1 2 3 4 5 6 7 8

R R R R R R R R

LED1 LED2 LED3 LED4 LED5 LED6 LED7 LED8

–1G –2G GND

1 19 10

74LS244
или 74HCT244

б)

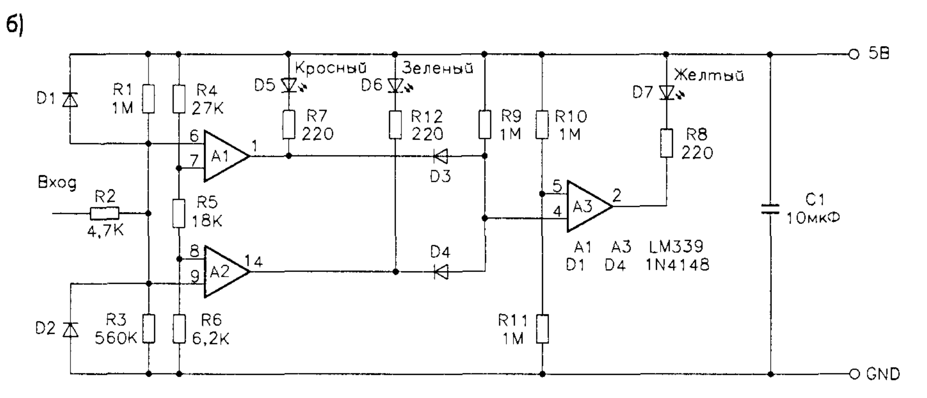
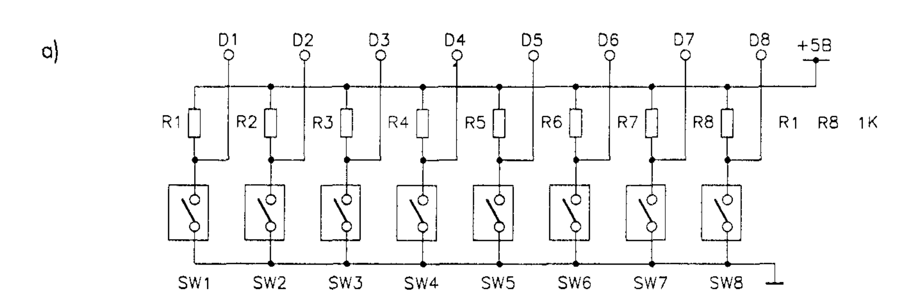


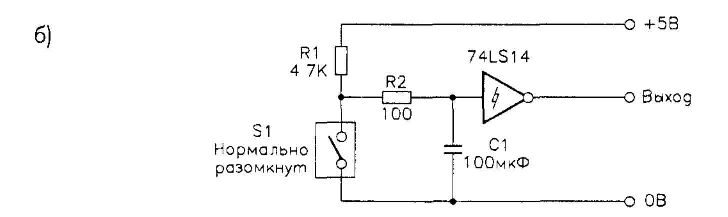
Рис. 2.13. Схемы логических пробников: а – на триггерах Шмитта 74LS244; б – на компараторе напряжений LM339

2.3.1. Цифровые генераторы сигналов

На рис. 2.14а показана схема восьмиканального *генератора логических состояний*. Он содержит восемь переключателей и резисторов по 1 кОм. Когда переключатель разомкнут, на выходе соответствующего канала формируется высокий уровень; при

его включении на выходе появляется низкий уровень. При высоком логическом уровне каждый канал может управлять 25 схемами ТТЛШ. Недостаток этой схемы состоит в так называемом *дребезге контактов*, когда при изменении положения переключателя состояние выходного сигнала изменяется не мгновенно, а сам сигнал содержит множество колебаний в течение короткого промежутка времени. Устранить подобное явление позволяет схема подавления дребезга контактов. На рис. 2.14б изображена такая схема с использованием триггера Шмитта с инвертором 74LS14. Когда переключатель замкнут, на выходе единица, когда разомкнут – ноль.





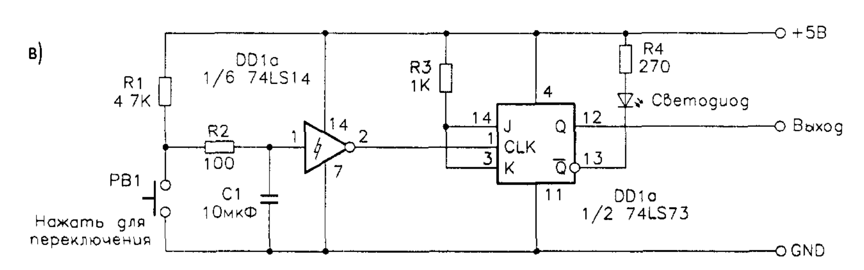


Рис. 2.14. Схемы генераторов логического состояния а – многоканальный генератор логических состояний, б – схема защиты от дребезга, в – переключатель состояния с защелкой

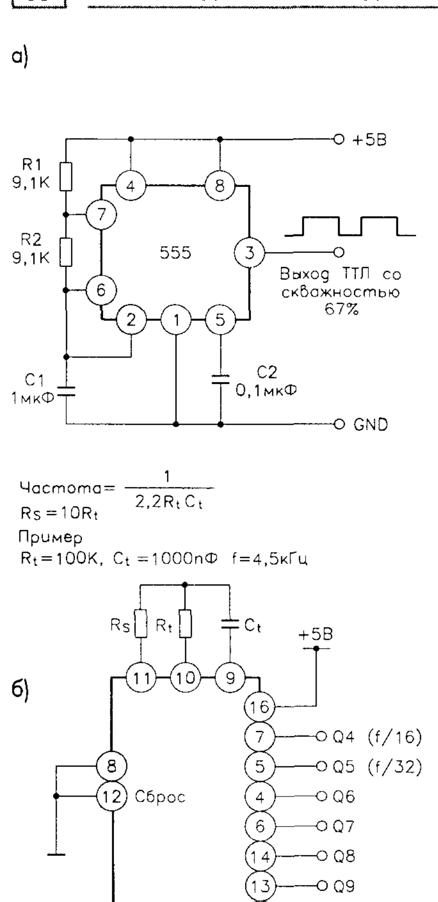


Рис. 2.15. Генераторы сигналов а – на таймере 555, б – на таймере CD4060

Другой тип логического генератора – это *генератор с защелкой* (рис. 2.14в). При нажатии на кнопку выходное состояние изменяется и остается постоянным до следующего нажатия.

Две схемы *генераторов прямоугольных импульсов* показаны на рис. 2.15. Первая схема построена на основе таймера (рис. 2.15а), она формирует сигнал частотой 1 кГц со скважностью 67%. На рис. 2.15б представлен генератор с перестраиваемой частотой, использующий 14-разрядный счетчик со встроенным генератором. Частота определяется резистором Rt и конденсатором Ct.

Для достижения большой точности частоты применяются *кварцевые генераторы*, три схемы которых представлены на рис. 2.16.

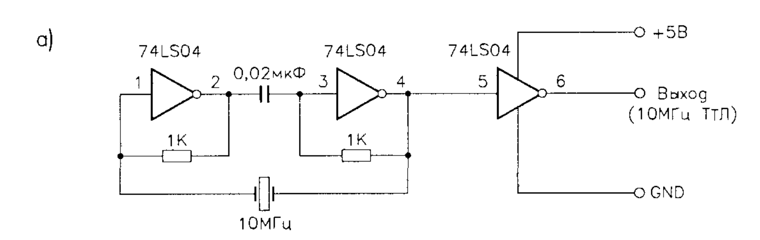
Первый генератор сигналов частотой 10 МГц использует опорный кварц 10 МГц и элементы инвертора 74LS04. Меньшие частоты можно получить с помощью делителей частоты. На рис. 2.16б изображена схема генератора на основе кварца 2,4576 МГц и счетчика со встроенным генератором CD4060, формирующим прямоугольные импульсы разных частот. На рис. 2.16в представлена схема программируемого кварцевого генератора EXO-3 (Interface Quartz Devices) с встроенным программируемым делителем частоты, который обеспечивает деление исходной частоты на 2^n (n = 1, 2, ..., 8).

Исходный сигнал формируется встроенным кварцевым генератором. Делитель выбирается с помощью выводов А, В и С. Номиналы исходных частот могут быть равны 12 (RS296-879), 14,318 (RS296-885), 18 (RS296-891) и 19,661 МГц (RS296-908).

2.3.2. Аналоговые генераторы сигналов

На рис. 2.17 изображен *аналоговый генератор сигналов* на базе микросхемы функционального генератора ICL8038BC (Harris Semiconductor). Он одновременно

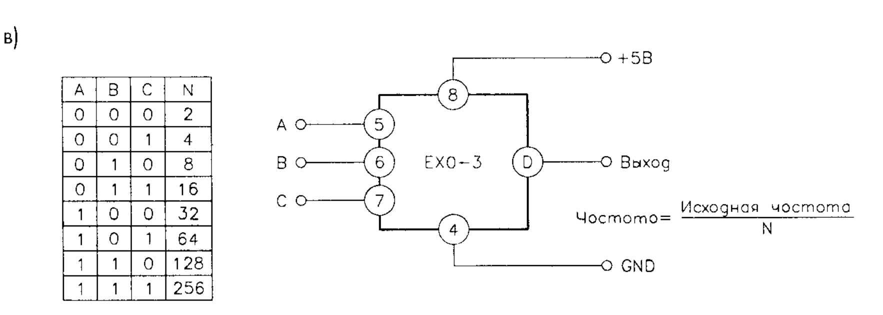
а)



б)

C1
C2
C1=C2=22пФ
X=2,4576МГц
R
4,7М
X=1МГц
C1=C2=33пФ
R=10М
X=4,9152МГц
C1=C2=27пФ
R=1М
+5В
10
11
16
7 153,6кГц
5 76,8кГц
8
12 Сброс
4 38,4кГц
6 19,2кГц
14 9,6кГц
13 4,8кГц
15 2,4кГц
1 600Гц
2 300Гц
CD4060
74HCT4060
3 150Гц

в)



A	B	C	N
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Рис. 2.16. Кварцевые генераторы: а – кварцевый генератор на микросхеме 74LS04; б – цифровой генератор на микросхеме CD4060; в – программируемый кварцевый генератор

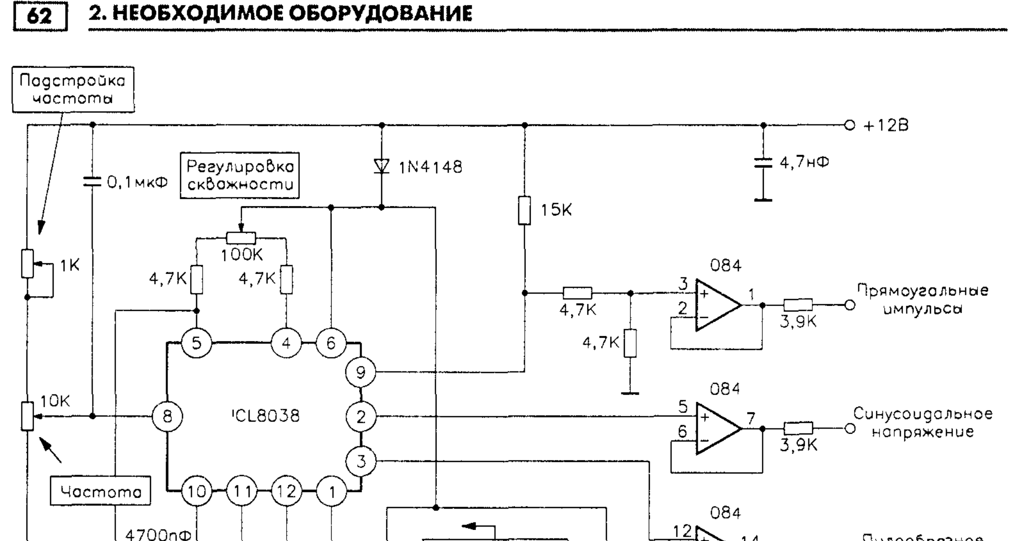


Рис. 2.17. Аналоговый генератор сигналов на ICL8038BC

формирует сигналы синусоидальной, прямоугольной и треугольной форм с перестраиваемой частотой в диапазоне 20–18000 Гц, которые через три операционных усилителя поступают на выход.

2.4. Экспериментальные платы параллельного, последовательного и игрового портов

Экспериментальные платы служат связующим звеном между компьютером и пользовательскими схемами. Они снабжены светодиодами для индикации логического состояния каждого входа и выхода, что улучшает наглядность операции ввода/вывода. Кроме того, на каждой плате имеется колодка, к которой можно подключить экспериментальную схему пользователя. Все входные цифровые линии подсоединяются к компьютеру через триггер Шмитта. Для работы плат необходим нестабилизированный источник постоянного тока 8–15 В. Размещенный на платах стабилизатор 7805 преобразует входное напряжение в напряжение +5 В. Ток источника питания ограничивается с помощью встроенного предохранителя на 1 А. В платах использованы распространенные электронные компоненты, монтаж выполнен на односторонних печатных платах, позволяющих проводить различные эксперименты по сопряжению внешних устройств с компьютером.

2.4.1. Экспериментальная плата параллельного порта

На рис. 2.18 изображена принципиальная схема экспериментальной платы параллельного порта.

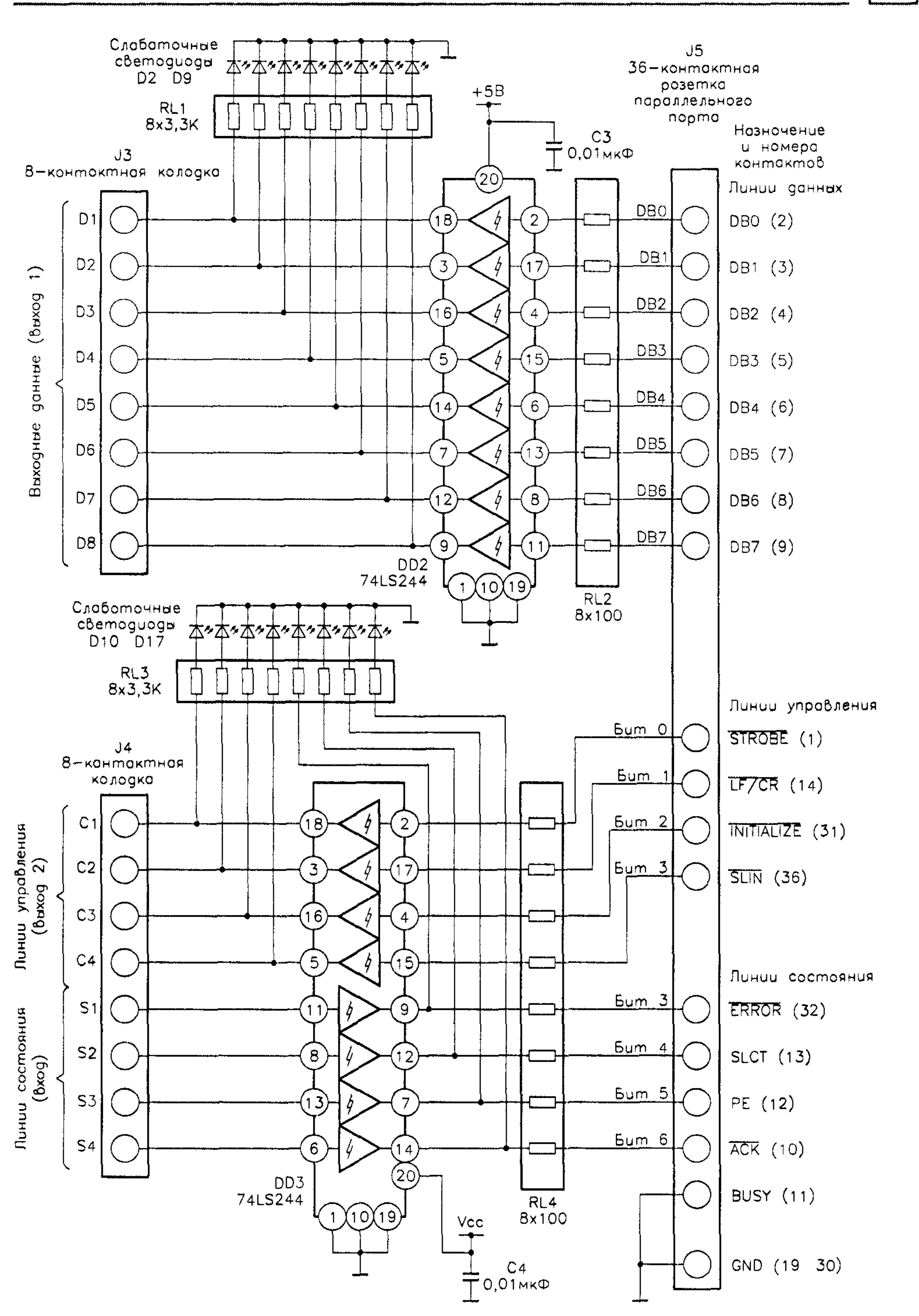


Рис. 2.18. Схема экспериментальной платы параллельного порта

Линии данных DB0 – DB7 порта заведены на входы буферов на триггерах Шмитта 74LS244 (DD2) через резистивную матрицу сопротивлением 100 Ом. Выходы буферов подключены к выходной колодке. Каждая линия соединена со светодиодом через резистор 3,3 кОм. Когда на линии высокий уровень, загорается соответствующий светодиод. Четыре выходные линии управления организованы на плате так же, как и линии данных. Четыре входа соединяются с четырьмя буферами DD3. Выходы буферов подключены к четырем входным линиям порта состояния через резисторы сопротивлением 100 Ом. Линии состояния порта имеют пять входных линий, но только четыре из них включены указанным образом. Логическое состояние таких линий также контролируется светодиодами.

Пятая линия порта состояния (BUSY) постоянно соединена с «землей». Это удобно, если для работы с платой используются команды управления принтером высокого уровня. На линии в таком случае устанавливается низкий уровень, показывающий, что плата готова к приему данных.

Источник питания выполнен на стабилизаторе 7805 (его необходимо установить на радиаторе) и имеет выходное напряжение +5 В при токе.1 А (рис. 2.19). Необходимые компоненты для сборки платы представлены в табл. 2.1.

Питание подается на плату через контакт SK1. Контакт SW1 управляет включением/выключением питания. Для защиты по току используется предохранитель. Состояние «включено/выключено» контролируется светодиодом. Входное и выходное напряжения подаются на четырехконтактную колодку J1.

Таблица 2.1. Компоненты, используемые в экспериментальной плате параллельного порта

Резисторы (металлопленочные, 1%, 0,25 Вт)	
R1	390 Ом
RL1, RL3	Восьмиканальная резистивная матрица, 3,3 кОм
RL2, RL4	Восьмиканальная резистивная матрица, 100 Ом
Конденсаторы	
C1, C3, C4	100 нФ
C2	10 мкФ
Полупроводниковые элементы	
DD1	7805 – стабилизатор напряжения, 1 А, +5 В
DD2, DD3	74LS244
D1	Зеленый светодиод, 5 мм
D2 – D17	Маломощные красные светодиоды, 3 мм
Разъемы и контакты	
J1	Четырехконтактная колодка
J2	Держатель для предохранителя
J3, J4	Восьмиконтактная колодка

Таблица 2.1. Компоненты, используемые в экспериментальной плате параллельного порта (окончание)

Разъемы и контакты	
J5	36-контактный гнездовой разъем параллельного порта
SK1	Штыревой разъем питания, 2,5 мм
Другие элементы	
SW1	Микропереключатель для печатных плат SPDT
Предохранитель	25 мм, 1 А
Радиатор	5 °С на ватт
Печатные платы	
Держатели для светодиодов	3 мм
Винты крепления печатных плат	

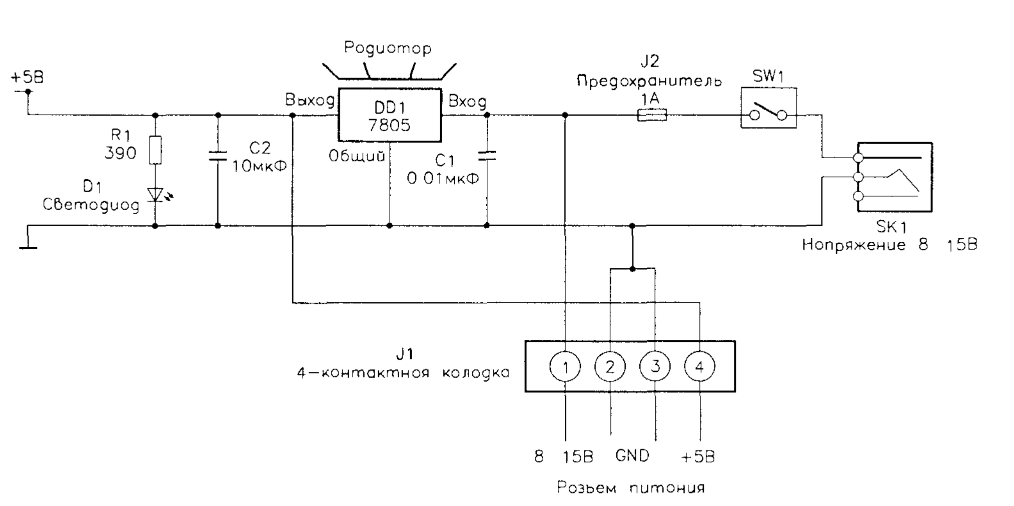


Рис. 2.19. Схема источника питания для плат сопряжения

2.4.2. Экспериментальная плата последовательного порта

Принципиальная схема экспериментальной платы последовательного порта приведена на рис. 2.20.

Три выходные линии последовательного порта (TD, RTS и DTR) подаются на приемники RS232-ТТЛ MAX238 (DD3, Maxim), где уровень напряжения RS232 преобразуется в уровень напряжения ТТЛ. Выходы DD3 соединены с буферами на триггерах Шмитта 74LS244 (DD2), выходы которых, в свою очередь, подключены к восьмиконтактной колодке J3. Логическое состояние каждой линии

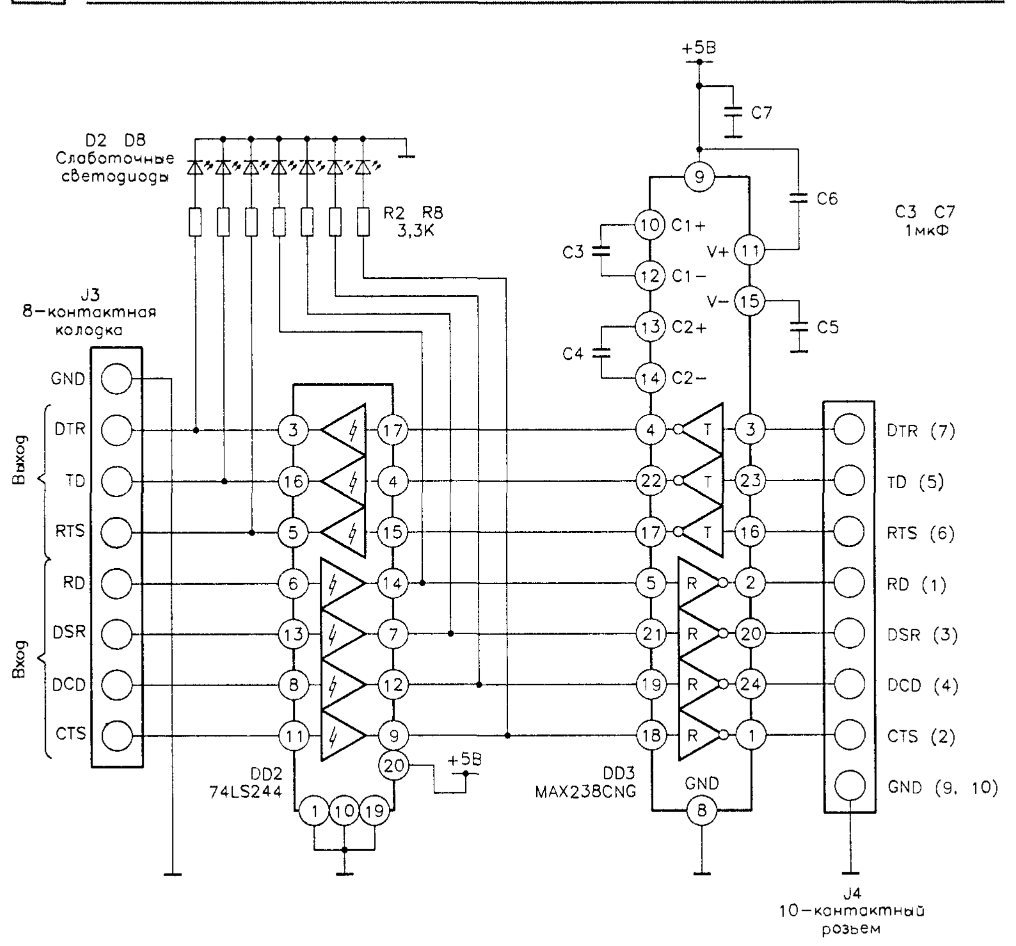


Рис. 2.20. Схема экспериментальной платы последовательного порта

контролируется при помощи светодиодов. Четыре входных сигнала (RD, DSR, DCD и CTS) подаются на буферы 74LS244. Выходы буферов соединены с DD3, где уровень напряжения ТТЛ преобразуется в уровень RS232; их логическое состояние контролируется светодиодами. Используется тот же источник питания, что и для экспериментальной платы параллельного порта.

Микросхема MAX238 – это преобразователь напряжения. Она содержит восемь преобразователей уровня, четыре из которых превращают уровень напряжения RS232 в уровень ТТЛ/КМОП, а четыре других – наоборот. Все преобразователи инвертируют сигнал. Для работы микросхемы необходимы пять внешних конденсаторов по 1 мкФ. Напряжение питания микросхемы составляет +5 В. Используемые компоненты приведены в табл. 2.2.

Таблица 2.2. Компоненты, используемые в экспериментальной плате последовательного порта

Резисторы (металлопленочные, 1%, 0,25 Вт)	
R1	390 Ом
R2 – R8	3,3 кОм
Конденсаторы	
C1	100 нФ
C2	10 мкФ
C3 – C7	1 мкФ
Полупроводниковые элементы	
DD1	7805 – стабилизатор напряжения, 1 А, +5 В
DD2	74LS244
DD3	MAX238CNG
D1	Зеленый светодиод, 5 мм
D2 – D8	Маломощные красные светодиоды, 3 мм
Разъемы и контакты	
J1	Четырехконтактная колодка
J2	Держатель для предохранителя
J3	Восьмиконтактная колодка
J4	Десятиконтактный разъем для печатной платы
SK1	Штыревой контакт питания, 2,5 мм
Другие элементы	
SW1	Микропереключатель для печатных плат SPDT
Предохранитель	25 мм, 1 А
Радиатор	5 °С на ватт
Печатные платы	
Держатели для светодиодов	3 мм
Винты крепления печатных плат	
Девятиконтактный гнездовой разъем D-типа в корпусе	
Девятижильный сигнальный кабель	1 м

2.4.3. Экспериментальная плата игрового порта

Принципиальная схема экспериментальной платы игрового порта приведена на рис. 2 21.

Четыре аналоговых входа (R1 – R4) подсоединены к четырем контактам восьмиконтактной колодки J3; напряжение +5 В от компьютера подводится к ней через предохранитель на 25 мА. Это напряжение используется не как напряжение

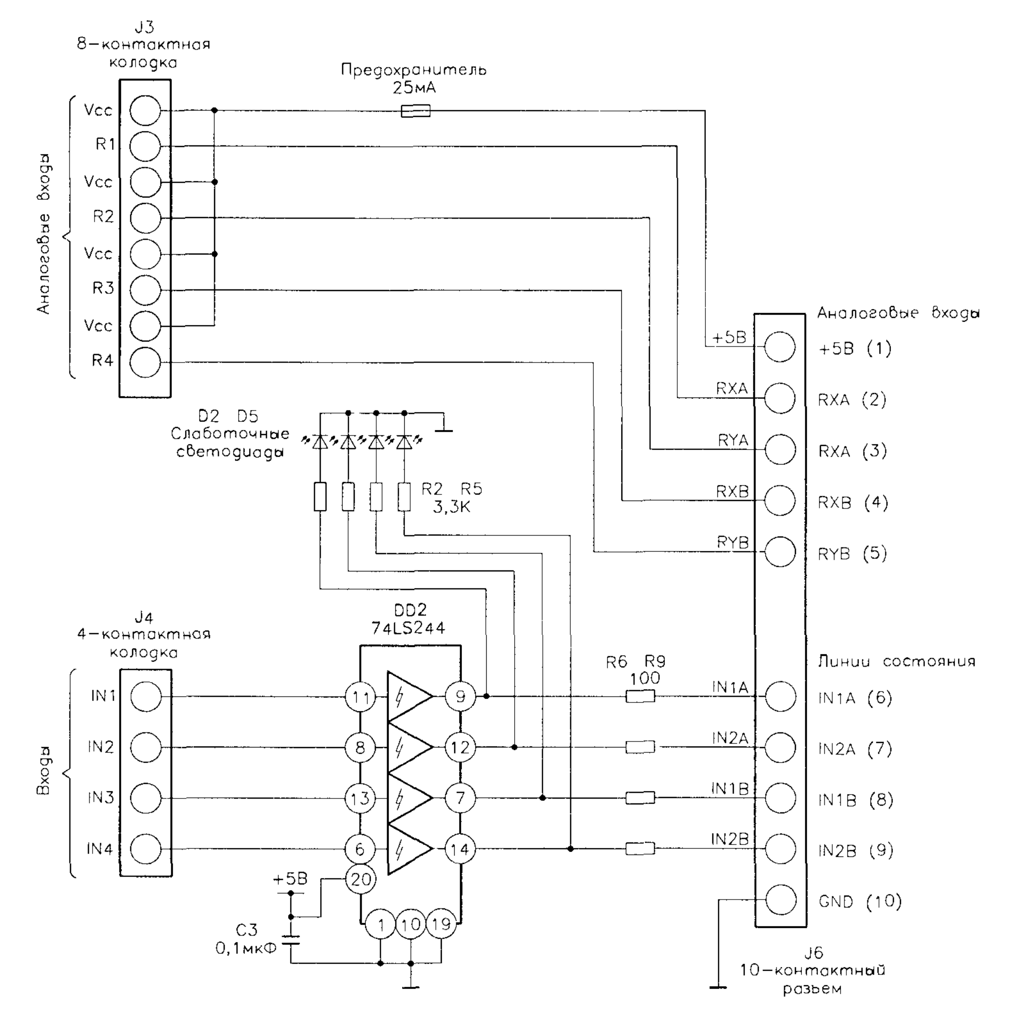


Рис. 2.21. Схема экспериментальной платы игрового порта

питания, а только для подключения резисторов. Сигналы с четырех цифровых входов (IN1 – IN4) поступают на буферы, выполненные на триггерах Шмитта 74LS244 (DD2). Выходы буферов соединены с четырьмя входными линиями игрового порта через резисторы по 100 Ом. Логические состояния цифровых входов отображаются светодиодами. Источник питания тот же, что и для экспериментальной платы параллельного порта. Необходимые компоненты приведены в табл. 2.3.

Игровые порты некоторых компьютеров поддерживают только один джойстик. В таком случае используются два аналоговых и два цифровых канала.

Таблица 2.3. Компоненты, используемые в экспериментальной плате игрового порта

Резисторы (металлопленочные, 1%, 0,25 Вт)	
R1	390 Ом
R2 – R5	3,3 кОм
R6 – R9	100 Ом
Конденсаторы	
C1, C3	100 нФ
C2	10 мкФ
Полупроводниковые элементы	
DD1	7805 – стабилизатор напряжения, 1 А, +5 В
DD2	74LS244
D1	Зеленый светодиод, 5 мм
D2 – D8	Маломощные красные светодиоды, 3 мм
Разъемы и контакты	
J1	Четырехконтактная колодка
J2	Держатель для предохранителя
J3	Восьмиконтактная колодка
J4	Четырехконтактная колодка
J5	Держатель для предохранителя
J6	Десятиконтактный разъем для печатной платы
SK1	Штыревой контакт питания, 2,5 мм
Другие элементы	
SW1	Микропереключатель для печатных плат SPDT
Предохранитель 1	25 мм, 1 А
Предохранитель 2	25 мм, 25 мА
Радиатор	5 °C на ватт
Печатные платы	
Держатели для светодиодов	3 мм
Винты крепления печатных плат	
15-контактный штыревой разъем D-типа в корпусе	
Десятижильный экранированный сигнальный кабель	1 м

2.4.4. Устройство экспериментальных плат

Фотошаблоны печатных плат изображены на рис. 2.22–2.24, расположение элементов показано на рис. 2.25–2.27.

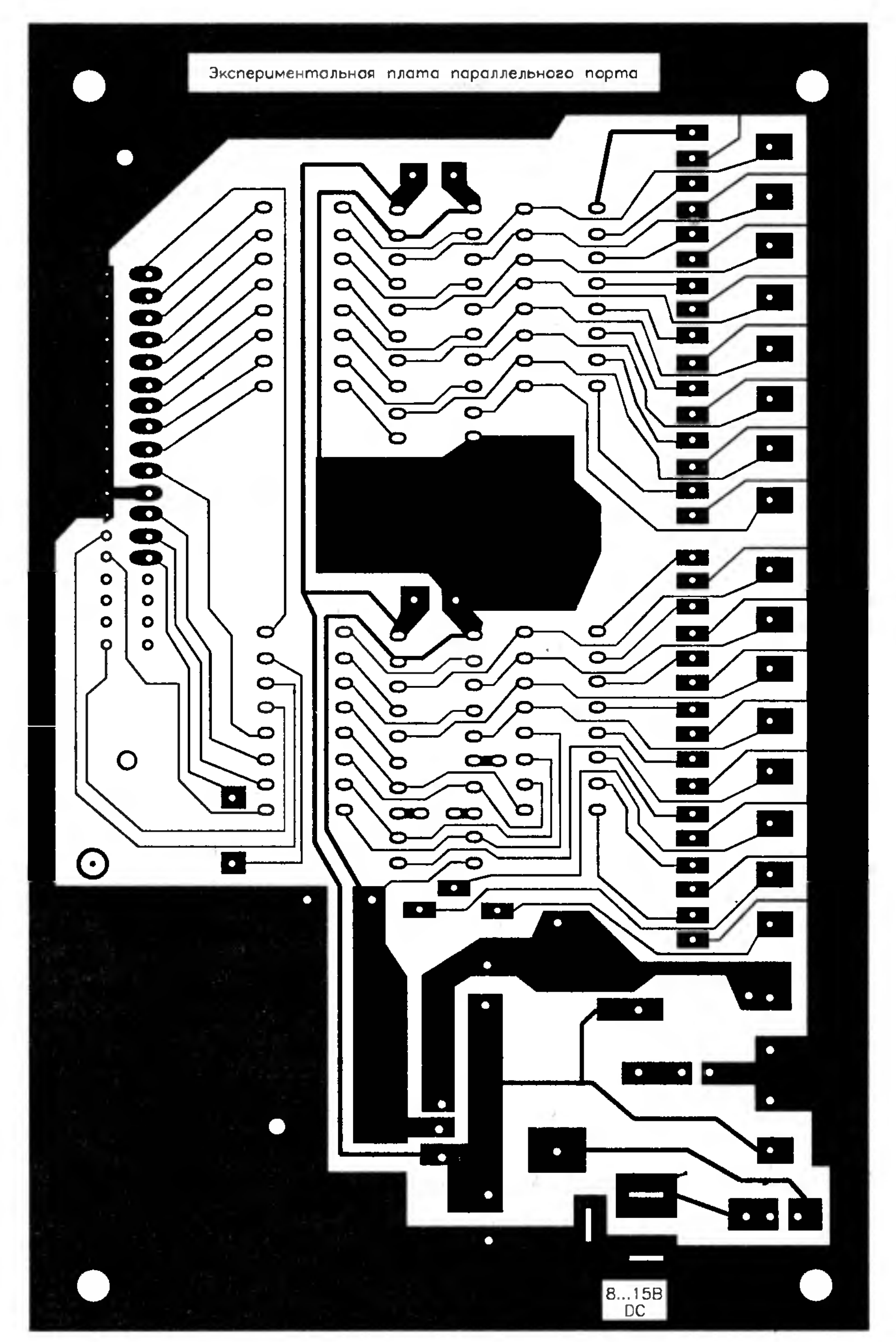


Рис. 2.22. Фотошаблон печатной платы для параллельного порта

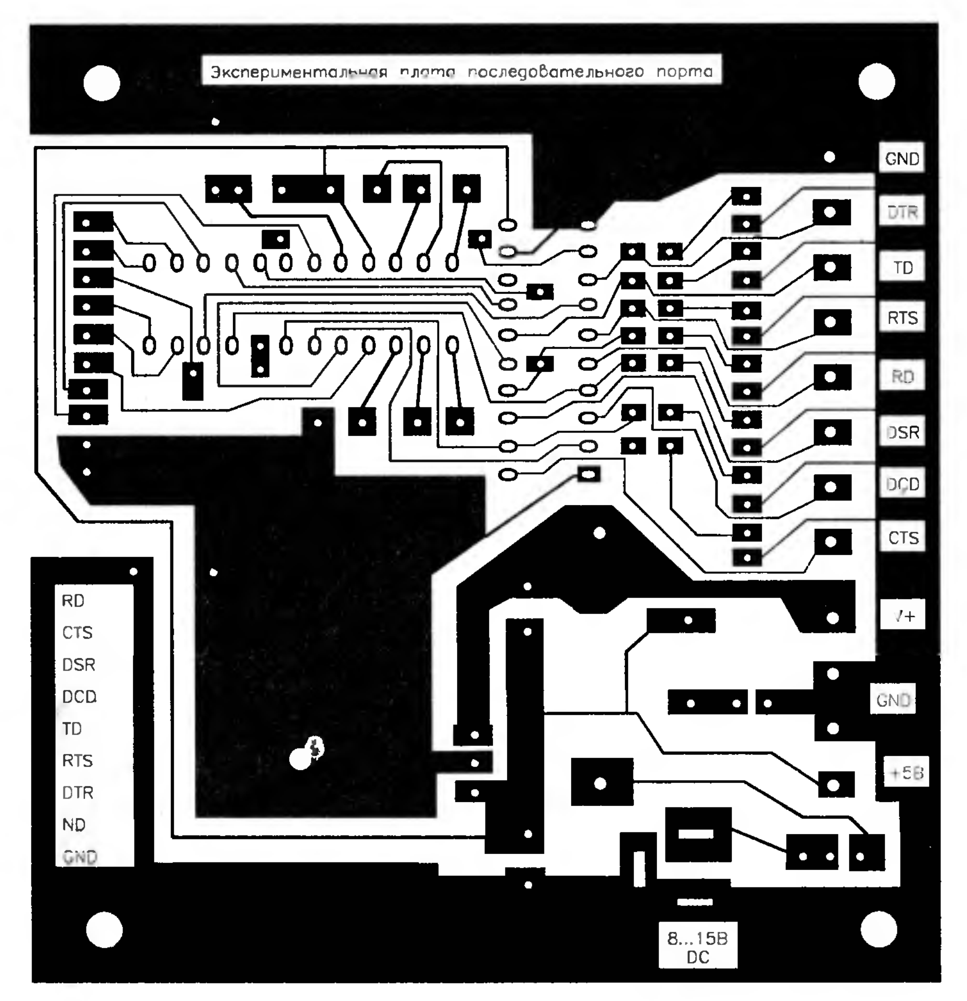


Рис. 2.23. Фотошаблон печатной платы для последовательного порта

2.5. Средства разработки плат

Средства разработки включают в себя макетные, монтажные и печатные платы Использование макетных плат – самый быстрый способ составления временных схем, поэтому он удобен на этапе экспериментальных работ Монтажные и печатные платы применяются для устройств, собранных по уже апробированным схемам

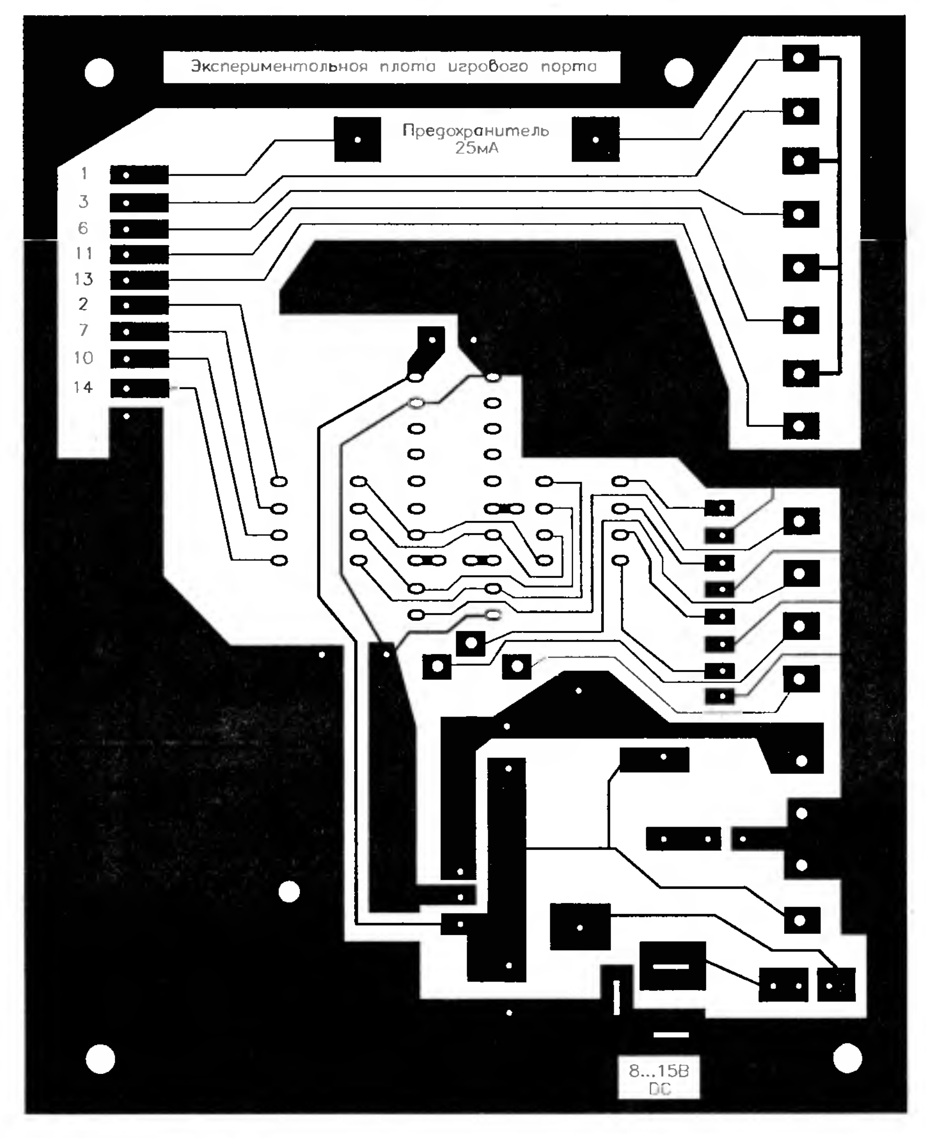


Рис. 2.24. Фотошаблон печатной платы для игрового порта

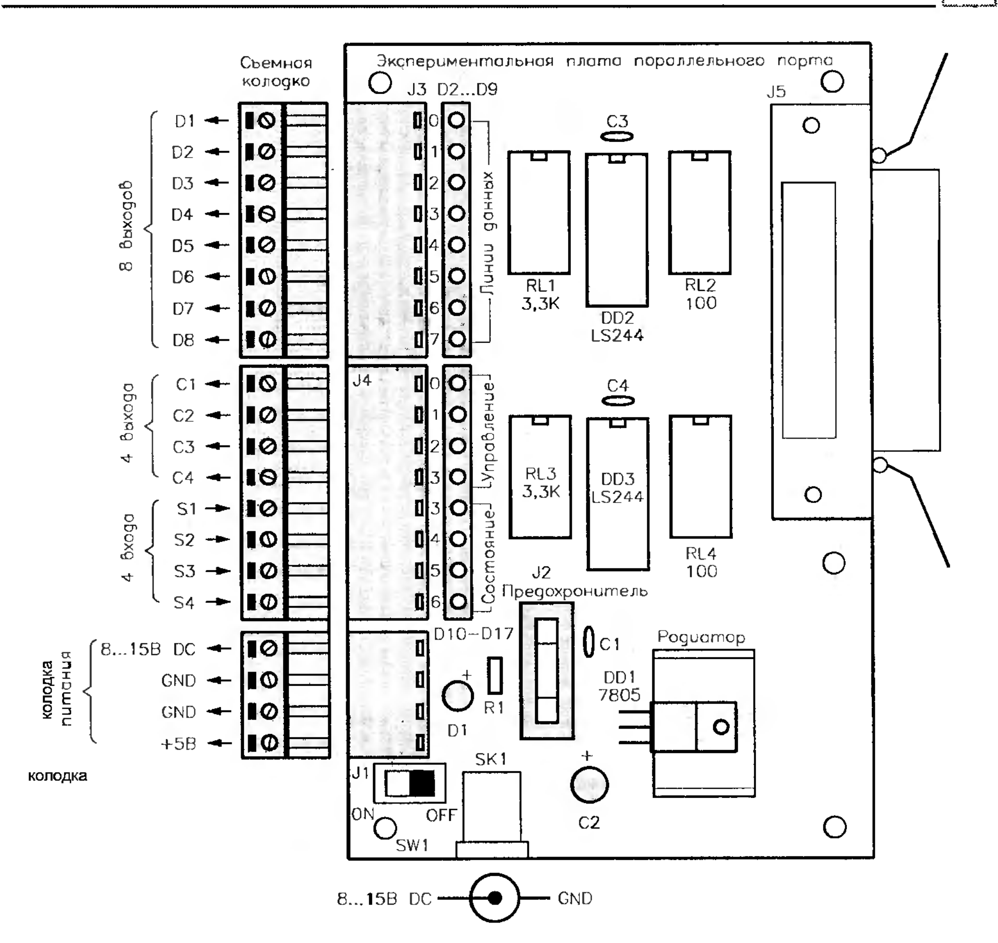


Рис. 2.25. Расположение элементов на экспериментальной плате параллельного порта

Рис. 2.26. Расположение элементов на экспериментальной плате последовательного порта

Рис. 2.27. Расположение элементов на экспериментальной плате игрового порта

3. ПРОГРАММЫ УПРАВЛЕНИЯ ЭКСПЕРИМЕНТАЛЬНЫМИ ПЛАТАМИ

В данной главе представлены программы управления экспериментальными платами параллельного, последовательного и игрового портов, написанные на трех языках: Borland Turbo Pascal 6 для DOS (TP6), Borland Turbo Pascal для Windows (TPW) и Microsoft Visual Basic 3 (VB3). Полный пакет ПО состоит из двух частей: непосредственно программ управления и библиотечных программ.

К числу программ управления относятся:

- CENTEXP.PAS для экспериментальной платы параллельного порта, язык TP6;
- CENTEXP для экспериментальной платы параллельного порта, язык VB3;
- RS232EXP.PAS для экспериментальной платы последовательного порта, язык TP6;
- RS232EXP для экспериментальной платы последовательного порта, язык VB3;
- GAMEEXP.PAS для экспериментальной платы игрового порта, язык TP6;
- GAMEEXP для экспериментальной платы игрового порта, язык VB3.

Библиотечных программ три:

- TPLIB1.PAS – библиотека ресурсов № 1, язык TP6;
- TPLIB2.PAS – библиотека ресурсов № 2, язык TP6;
- WLIB1.PAS – библиотека ресурсов № 1, язык TPW.

Библиотеки ресурсов, написанные на языке TP6, имеют набор процедур и функций для основных операций ввода/вывода параллельного, последовательного и игрового портов, обработки нажатия клавиши, вывода сообщений на экран и т.д. Эти библиотеки можно включить в пользовательские программы, тогда все процедуры и функции вызываются в дальнейшем из них. Те же программы, написанные на TPW, допустимо оформить в виде библиотеки динамической компоновки для

Windows (DLL) и впоследствии вызывать из любой программы для Windows, написанной на VB3, Visual C и т.д.

Использование программ позволяет проследить основные операции ввода/вывода и провести простые эксперименты по сопряжению компьютера с внешними устройствами.

3.1. Программное обеспечение для экспериментальной платы параллельного порта

В разделе представлены тексты программ управления экспериментальной платой параллельного порта и необходимые пояснения к ним.

3.1.1. Описание программы CENTEXP.PAS

Программа выполняет следующие функции:

- сообщает о количестве установленных на компьютере параллельных портов;
- позволяет выбрать номер порта для дальнейшего использования;
- изменяет состояние битов регистра данных (8 бит) и регистра управления (4 бита);
- считывает данные из регистра состояния (4 бита).

После запуска программы на экран выводится следующая информация:

```
Number of LPT installed  2
Addresses for LPT1 to LPT4  888 632 0 0
Select LPT to be used (1,2,3 or 4)
```

Первая строка показывает количество установленных на компьютере параллельных портов (в данном примере – два), вторая – их базовые адреса. Третья строка позволяет выбрать номер порта для дальнейшего использования, при этом на экран выводится виртуальная панель управления (рис. 3.1).

На панели изображено 16 виртуальных светодиодов, соответствующих реальным светодиодам на экспериментальной плате. Восемь из них соотносятся с битами регистра данных, четыре – с битами регистра управления и оставшиеся четыре – с битами регистра состояния. Внизу экрана расположена панель помощи, на которой показаны клавиши управления программой с кратким описанием их функций:

- **[ARROW]** (клавиши управления курсором) – выбор выходной линии;
- **[SPASE]** (пробел) – изменение состояния выбранной линии;
- **[Q]** или **[q]** – выход из программы.

Нажимая на клавиши ← и →, можно указать одну из 12 линий, которые управляются компьютером. Выбранная линия помечается окружностью вокруг виртуального светодиода. Состояние светодиода изменяется нажатием на клавишу пробела; красный цвет свидетельствует о наличии высокого уровня.

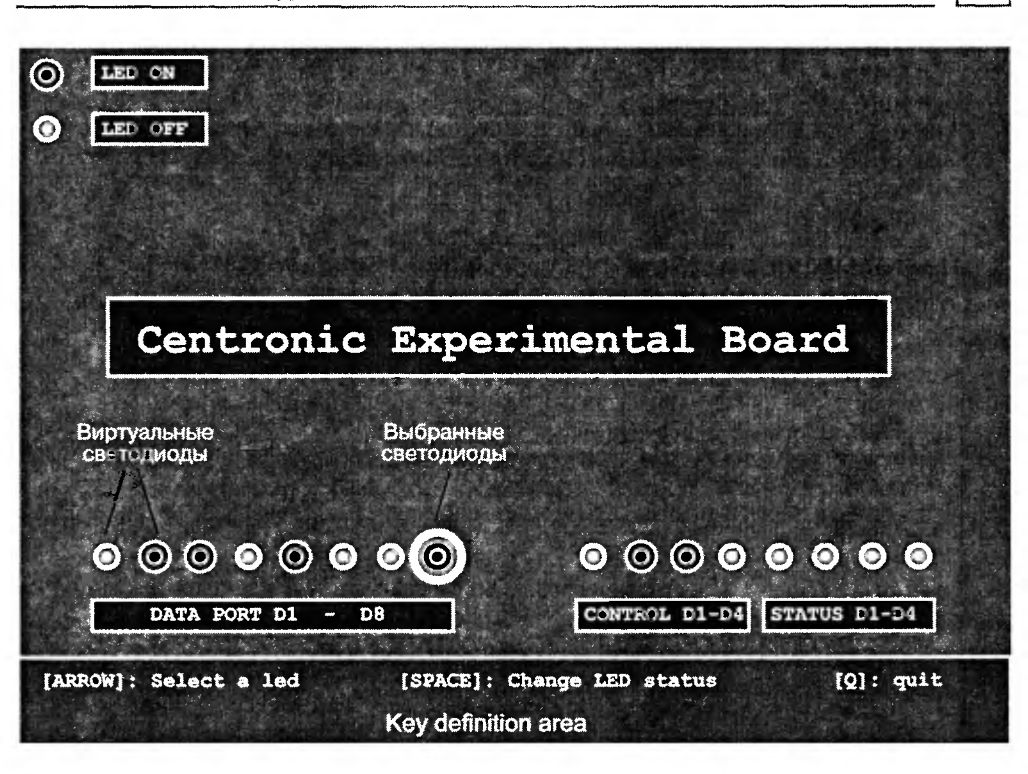


Рис. 3.1. Виртуальная панель управления для экспериментальной платы параллельного порта

Текст программы CENTEXP.PAS

```
Program Centronic_Experimental_Board.
(*Программа управления экспериментальной платой параллельного порта *)
uses
  graph,crt,dos;
var
  i.led_selected integer;
  ch char;
  status.array[1..18] of integer,
  key_pressed:string[10];

(*Подключение двух библиотек: TLIB1 и TLIB2.*)
{$I c \ioexp\tplib1.pas}
{$I c \ioexp\tplib2.pas}

procedure Draw_panel;
(*Рисование панели управления экспериментальной платой.*)
begin
  (*Рисование 16 светодиодов на экране *)
  setbkcolor(cyan);
```

```
   for i:=1 to 16 do status[i]:=0;
   for i:=1 to 8 do draw_led(30+i*30, 350, status[i]);
   for i:=1 to 8 do draw_led(340+i*30, 350, status[8+i]);
   (*Вывод названий.*)
   draw_led(20,20,1); draw_message(50,20,70,20,lightblue,'LED ON',0,1,yellow);
   draw_led(20,60,0); draw_message(50,60,70,20,lightblue,'LED OFF',0,1,yellow);
   draw_message(50,390,230,20,blue,' DATA PORT D1 - D8 ',0,1,yellow);
   draw_message(360,390,110,20,blue,' CONTROL D1 - D4 ',0,1,yellow);
   draw_message(480,390,110,20,blue,' STATUS D1 - D4 ',0,1,yellow);
   (*Рисование панели помощи внизу.*)
   setfillstyle(1,magenta);
   bar(1,420,800,480);
   settextstyle(0,0,1);
   outtextxy(20,430,'[ARROW]: Select a LED   [SPACE]:Change LED status   [Q]:Quit');
   (*Рисование центрального сообщения.*)
   draw_message(60,200,500,50,blue,'Centronic Experimental Board',0,1,yellow);
   (*Инициализация выходов.*)
   write_data_port(p_address,0);
   write_control_port(P_address,0);
end;

Procedure Output_Input;
(*Процедура ввода/вывода.*)
var
   output_byte, input_byte:byte;
begin
   (*Вычисление значения данных, передаваемых в порт данных.*)
   output_byte:=0;
   for i:=1 to 8 do output_byte:=output_byte+status[i]*bit_weight(i);
   write_data_port(P_address,output_byte);
   (*Вычисление значения данных, передаваемых в порт управления.*)
   output_byte:=0;
   for i:=9 to 12 do output_byte:=output_byte+status[i]*bit_weight(i-8);
   write_control_port(P_address,output_byte);
   (*Считывание данных из порта состояния и вычисление состояния светодиодов.*)
   input_byte:=read_status_port(P_address);
   for i:=1 to 4 do status[12+i]:=round((input_byte and bit_weight(i))/bit_weight(i));
end;

Procedure scan_keyboard;
(*Опрос клавиатуры для обнаружения нажатия клавиши.*)
var
   led_selected_old:integer;
begin
   led_selected_old:=led_selected;
   (*Обнаружение нажатия клавиши.*)
   key_pressed:=getkey;
   if key_pressed='LEFT' then led_selected:=led_selected-1;
   if key_pressed='RIGHT' then led_selected:=led_selected+1;
   if key_pressed=' ' then status[led_selected]:=1-status[led_selected];
   (*Показ виртуальных светодиодов и их состояния.*)
   setbkcolor(cyan);
   for i:=1 to 8 do draw_led(30+i*30,350,status[i]);
```

```
    for i:=1 to 4 do draw_led(340+i*30,350,status[8+i]);
    output_input;
    for i:=5 to 8 do draw_led(340+i*30,350,status[8+i]);
    if led_selected>12 then led_selected:=12;
    if led_selected<1 then led_selected:=1;
    (*Отображение окружности вокруг выбранного светодиода.*)
    setlinestyle(0,0,3);
    setcolor(cyan);
    if led_selected_old<=8 then circle(30+30*led_selected_old,350,15)
       else circle(340+30*(led_selected_old-8),350,15);
    setcolor(yellow);
    if led_selected<=8 then circle(30+30*led_selected,350,15)
       else circle(340+30*(led_selected-8),350,15);
end;

(*Главная программа.*)
begin
    centronic_address;      (*Ввод адреса параллельного порта.*)
    initialize_graph;       (*Инициализация графического режима.*)
    draw_panel;             (*Рисование виртуальной панели.*)
    led_selected:=1;
    repeat
       scan_keyboard;       (*Опрос клавиатуры для обнаружения нажатия клавиши.*)
    until (key_pressed='Q') or (key_pressed='q');
    closegraph;             (*Закрытие графического режима.*)
end.
```

Здесь используются две библиотеки – TPLIB1.PAS и TPLIB2.PAS, которые подключаются с помощью директивы TP6 INCLUDE.

```
{$I c:\ioexp\tplib1.pas}
{$I c:\ioexp\tplib2.pas}
```

Программа содержит три главные процедуры, находящиеся в библиотеке TLIB2.PAS: Draw_panel рисует виртуальную панель управления экспериментальной платой, Draw_led() и Draw_message() – изображения виртуальных светодиодов и надписи. Процедура Output_input управляет операциями ввода/вывода параллельного порта, Write_data_port(), write_control_port() и read_status_port() – операциями записи в регистры порта и чтения из них; они содержатся в библиотеке TPLIB1.PAS.

Процедура Scan_keyboard опрашивает клавиатуру для обнаружения нажатия клавиш управления. Здесь используется функция getkey из библиотеки TPLIB2.PAS.

3.1.2. Описание программы CENTEXP

Программа написана на языке VB3 и выполняет следующие функции:

- сообщает о количестве установленных на компьютере параллельных портов;
- позволяет выбрать номер порта для его дальнейшего использования;
- изменяет состояние битов регистра данных (8 бит) и регистра управления (4 бита);
- считывает данные из регистра состояния (4 линии).

Для запуска программы в среде Windows нужно нажать кнопку **Пуск** и выбрать пункт меню **Выполнить**, а затем вписать ее полное имя путь + CENTEXP. После щелчка по кнопке **OK** на экран выводится сообщение о количестве параллельных портов, установленных на компьютере, и их базовых адресах (рис. 3 2).

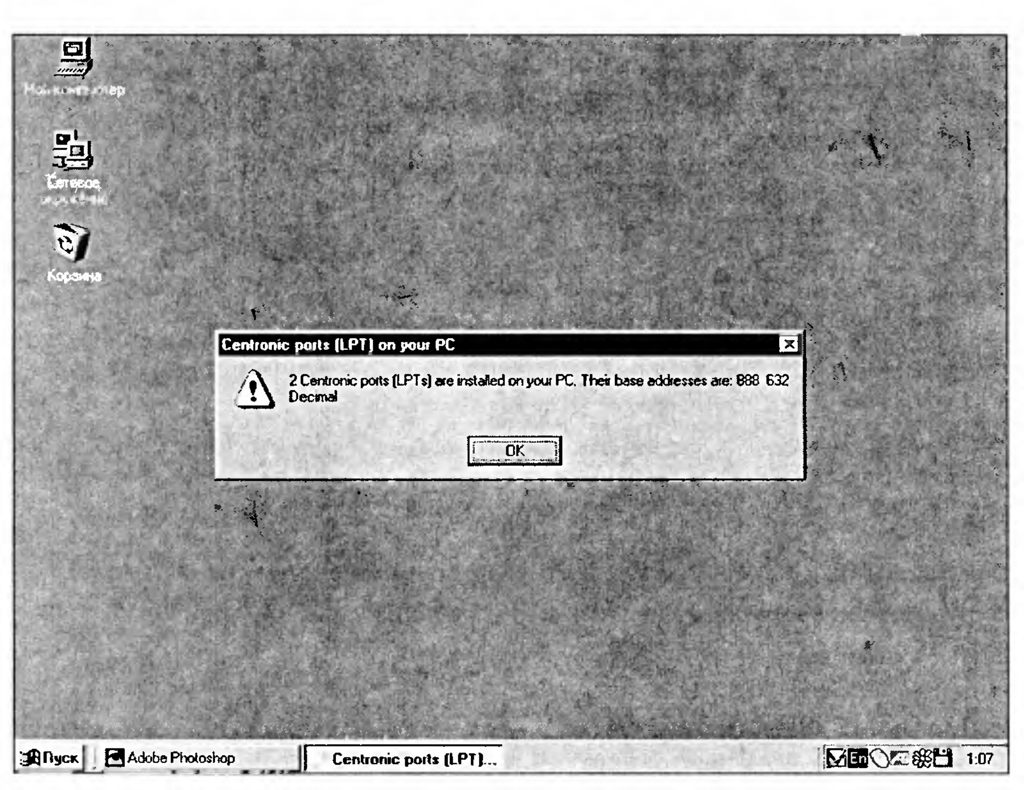


Рис. 3.2. Сообщение о количестве установленных параллельных портов

Нажмите кнопку **OK**, и на экране появится другое окно (рис. 3.3). Здесь необходимо выбрать номер порта, к которому подключена экспериментальная плата. Введите номер порта (1–4) и щелкните по **OK**. Вы увидите окно управления экспериментальной платой параллельного порта (рис. 3.4), содержащее информационную панель и 15 кнопок, которые выполняют следующие функции·

- **Get it** – обновление информации о состоянии входов. Красный цвет светодиода свидетельствует о высоком уровне соответствующей линии,
- **0-1** – изменение состояния битов регистра данных и управления При этом на панель выводится номер указанного порта и его базовый адрес,
- **Change** – выбор другого порта,
- **Quit** – выход из программы.

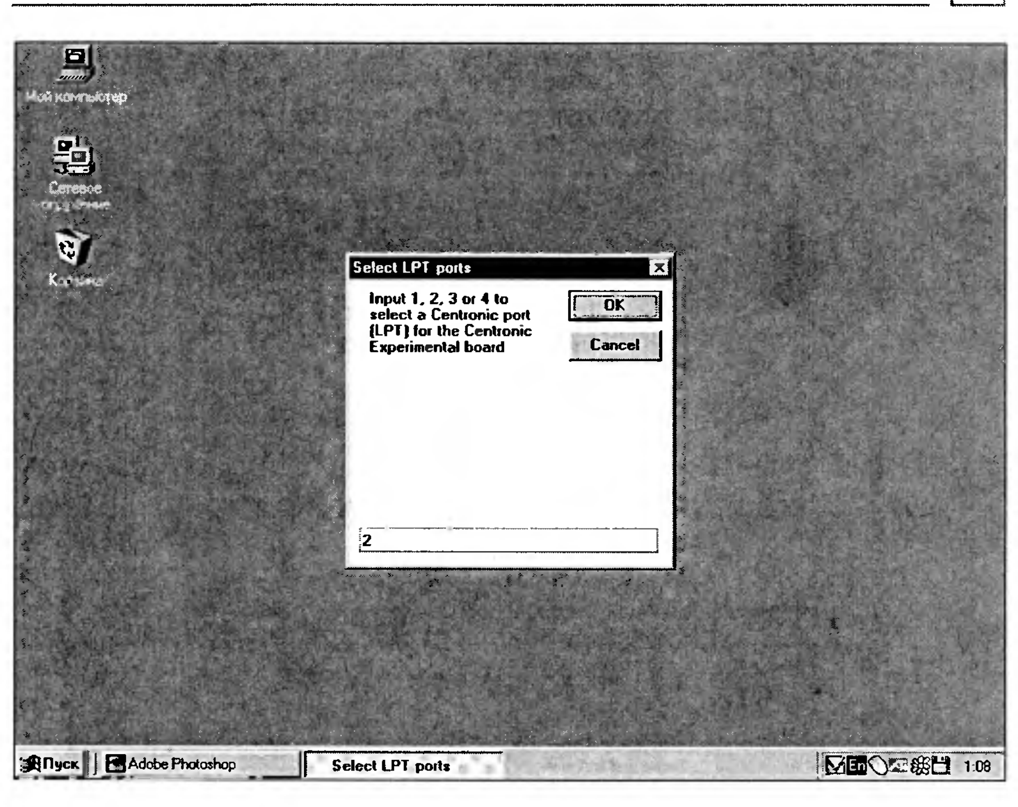


Рис. 3.3. Окно выбора параллельного порта

Текст программы CENTEXP

```
  Объявление функции библиотеки динамической компоновки WLIB1 DLL
  Объявляемые функции  Centronic()  Bit_weight()  Read_status_port()
                       Write_data_port() и Write_control_port()
Declare Function Centronic Lib  C \Ioexp\Wlib1 dll  (ByVal X As Integer) As Integer
Declare Function Bit_weight Lib  C \Ioexp\Wlib1 dll  (ByVal X As Integer) As Integer
Declare Function Read_status_port Lib  C \Ioexp\Wlib1 dll  (ByVal address As Integer) As
Integer
Declare Function Write_data_port Lib  C \Ioexp\Wlib1 dll  (ByVal address As Integer  ByVal
Output_data As Integer) As Integer
Declare Function Write_control_port Lib  C \Ioexp\Wlib1 dll  (ByVal address As Integer  ByVal
Output_data As Integer) As Integer

Sub Command1_Click()
  input_byte =read_status_port(P_address)
  For i =12 to 15
  status(i)=(input_byte And Bit_weight(i 11))/Bit_weight(i-11)
  If status(i)=1 Then Shape1(i) BackColor=&HFF& Else Shape1(i) BackColor=black
  Next i

End Sub
```

Рис. 3.4. Окно управления экспериментальной платой параллельного порта

```
Sub Command1_MouseMove(Button As Integer  Shift As Integer, X As Single  Y As Single)
  Label3 Caption= Get the status of the Status port inputs (4 inputs)
End Sub

Sub Command2_click (index As Integer)
   Изменение состояния выходов регистра данных и управления
  status(index)=1-status(index)                          Изменение
  If status(index) =1 Then Shape1(index) BackColor=&HFF& Else Shape1(index) BackColor=black
   Вывод состояния на виртуальные светодиоды
   Вывод байта в регистр данных
  Output_byte=0
  For i=0 To 7
  Output_byte=Output_byte+status(i)*Bit_weight(i+1)  Формирование байта для передачи
  Next i
  dummy=Write_data_port(P_address Output_byte)       Запись байта в регистр данных

   Вывод баита в регистр управления
  Output_byte=0
  For i=8 To 11
  Output_byte=Output_byte+status(i)*Bit_weight(i-7)  Формирование байта для передачи
  Next i
  dummy=Write_control_port(P_address Output_byte)    Запись байта в регистр управления
```

```
   'Чтение данных регистра состояния.
   input_byte=Read_status_port(P_address)
   For i=12 To 15
   'Определение состояния каждого бита.
   status(i)=(input_byteAnd Bit_weight(i-11))/Bit_weight(i-11)
   'Вывод состояния на виртуальные светодиоды.
   If status(i)=1 Then Shape1(i).BackColor=&HFF& Else Shape1(i).BackColor=black
   Next i
End Sub

Sub Command2_MouseMove(index As Integer, Button As Integer, Shift As Integer, X As Single,
Y As Single)
   'Отображение подсказки при движении указателя мыши над кнопкой.
   If index<=7 Then
   Label3.Caption="Change status of Data port outputs (8 outputs)"
   Else
   Label3 Caption="Change status of Control port outputs (4 outputs)"
   End If
End Sub

Sub Command3_Click()
   'Выбор другого порта.
   dummy=MsgBox(Str(Centronic(0))-1 & "Centronic ports (LPTs) are installed on your PC. Their
base addresses are: " & Format$(Centronic(1), "###") & "  " & Format$(Centronic(2), "###") & "  "
& Format$(Centronic(3), "###") & "  " & Format$(Centronic(4), "###") & "Descimal",48,"Centronic
ports (LPT) on your PC")              'Отображение информации
                                      'о количестве установленных LPT-портов.
   lpt_number=Val(InputBox$("Input 1, 2, 3 or 4to select a Centronic port (LPT) for the Mini-
Lab Data Logger/Controller", "Select LPT ports"))   'Выбор параллельного порта.
   P_address=Centronic(lpt_number)                  'Нахождение базового адреса
                                                    'выбранного порта.
   Label2.Caption="Selected LPT port:" & Format(lpt_number)  'Отображение информации
                                                             'о выбранном LPT-порте.
   Label4.Caption="Base address of LPT:" & Format(P_address) 'Отображение информации
                                                             'об адресе выбранного порта.
End Sub

Sub Command4_Click()
   'Выход из программы.
   End
End Sub

Sub Command4_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label3.Caption="Quit the program"
End Sub

Sub Form_Load()
   'Инициализация состояния.
   For i=0 to 11
       status(i)=0
   Next i

   'Отображение информации о параллельном порте, выбор LPT-порта.
   dummy=MsgBox(Str(Centronic(0))-1 & "Centronic ports (LPTs) are installed on your PC.
Their base addresses are: " & Format$(Centronic(1),"###") & "  " &
```

```
Format$(Centronic(2),"###") & "  " & Format$(Centronic(3),"###") & "  " &
Format$(Centronic(4),"###") & "Descimal",48,"Centronic ports (LPTs) on your PC")
  lpt_number=Val(InputBox$("Input 1, 2, 3 or 4 to select Centronic port (LPT) for the
Centronic experimental board", "Select LPT ports"))
  P_address=Centronic(lpt_number)
  Label2.Caption="No of installed LPTs: " & Format(lpt_number)
  Label4 Caption="Base address of LPT: " & Format(P_address)
  dummy=Write_data_port(P_address,0)
  dummy=Write_control_port(P_address,0)
End Sub
```

3.2. Программное обеспечение для экспериментальной платы последовательного порта

В данном разделе представлены тексты программ управления экспериментальной платой последовательного порта и необходимые пояснения к ним.

3.2.1. Описание программы RS232EXP.PAS

Программа выполняет следующие функции:

- сообщает о количестве установленных на компьютере последовательных портов;
- позволяет выбрать последовательный порт (COM);
- настраивает формат последовательных данных;
- вводит байт и передает последовательные данные;
- изменяет состояние линий управления модемом (DTR и RTS);
- считывает последовательные данные;
- считывает состояние линий DSR, DCD и CTS.

После запуска программы на экран выводится следующая информация:

```
Number of COM installed. 4
Addresses for COM1 to COM4. 1016 760 1000 744
Select a COM to be used (1,2,3 or 4).
```

Первая строка показывает количество установленных COM-портов, вторая – отображает их базовые адреса. Третья строка дает возможность выбрать последовательный порт для его дальнейшего использования. После того как вы укажете номер порта, на экране появится виртуальная панель управления (рис. 3.5).

Здесь изображены семь виртуальных светодиодов в соответствии с их количеством на экспериментальной плате: три из них соотносятся с тремя выходами, четыре – с четырьмя входами. Внизу размещается панель помощи, где показаны клавиши управления программой с кратким описанием их функций:

- **[ARROW]** (клавиши управления курсором) – выбор выхода;
- **[T]** или **[t]** – ввод данных с клавиатуры и их передача;
- **[SPASE]** (пробел) – изменение состояния выбранной выходной линии;
- **[C]** или **[c]** – настройка формата последовательных данных;
- **[Q]** или **[q]** – выход из программы.

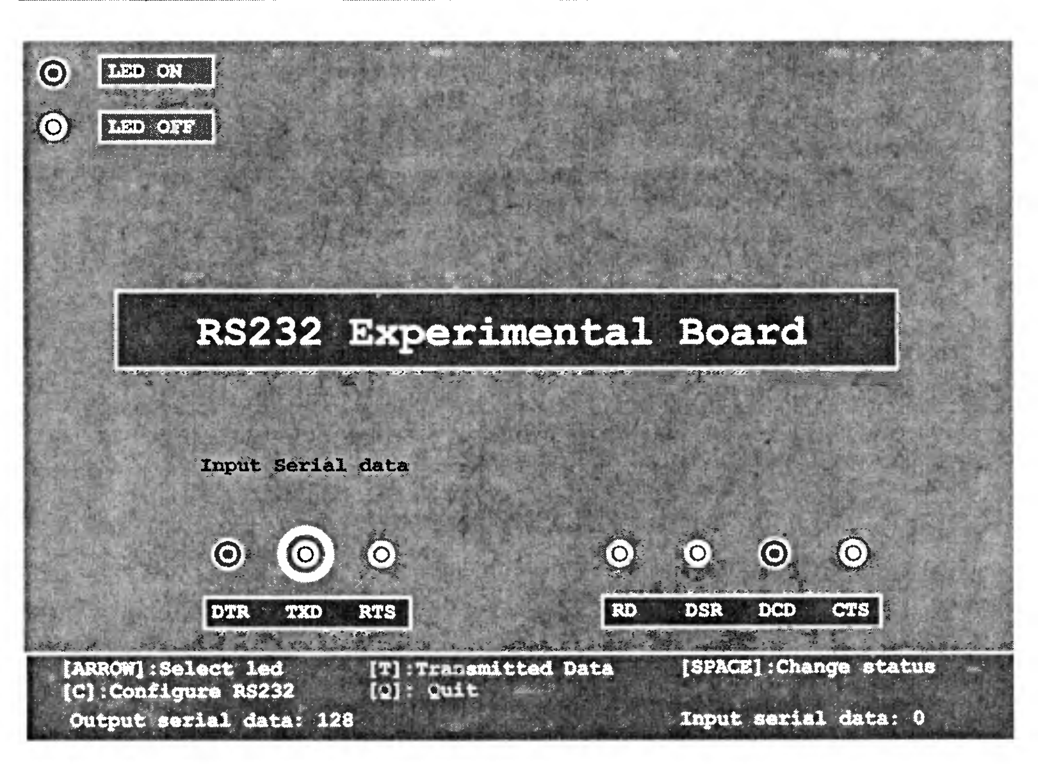


Рис. 3.5. Виртуальная панель управления для экспериментальной платы последовательного порта

Посредством клавиш ← и → можно указать одну из выходных линий. Выбранный выход помечается окружностью вокруг изображения светодиода. Для изменения состояния выхода используется клавиша пробела. При нажатии на клавиши →, ← или клавишу пробела входные данные считываются и выводятся на экран. Для тестирования порта нужно соединить приемную и передающую линии. Значения входных и выходных последовательных данных должны совпадать.

Текст программы RS232EXP.PAS

```
Program RS232_tester
(*Программа управления экспериментальной платой последовательного порта *)
uses
  graph crt,dos,
var
  i,led_selected Serial_input_byte,Baud_rate_byte
  data_length_byte,Stop_length_byte,Parity_byte integer
  Serial_output_string,Serial_input_string Old_serial_input_string   string[5]
  ch char,
  status array[1  18] of integer,
  key_pressed string[10]

(*Загрузка двух библиотек *)
{$I c \ioexp\tplib1 pas}
{$I c \ioexp\tplib2 pas}
```

```
procedure Draw_panel;
(*Рисование панели управления экспериментальной платой последовательного порта.*)
begin
  for i:=1 to 16 do status[i]:=0;
  setbkcolor(cyan);
  for i:=1 to 3 do draw_led(80+i*50,350,status[i]);
  for i:=4 to 7 do draw_led(180+i*50,350,status[i]);
  draw_led(20,20,1);
  draw_message(50,20,70,20,lightblue,'LED ON',0,1,yellow);
  draw_led(20,60,0);
  draw_message(50,60,70,20,lightblue,'LED OFF',0,1,yellow);
  setfillstyle(1,magenta);
  bar(1,420,800,480);
  settextstyle(0,0,1);
  outtextxy(10,425,'[ARROW]:Select LED [T]:Transmitted data     [Space]:Change Status');
  outtextxy(10,440.'[C]:Configure RS232     [Q]:Quit');
  draw_message(60,200,500,50,blue,'RS232 experimental board',0,2,yellow);
  draw_message(115,390,130,20,blue,'DTR          TXD RTS',0,1,yellow);
  draw_message(370,390,180,20,blue,'RD       DSR DCD CTS',0,1,yellow);
end;

procedure Output_input;
var
  output_byte,input_byte:byte;
  code:integer;
begin
  (*Вычисление значения данных для передачи через регистр данных.*)
  write_modem_status(RS232_address, status[3], status[1]);
  val(serial_output_string,output_byte,code);
  status[5]:=read_modem_status(RS232_address,2);
  status[6]:=read_modem_status(RS232_address,1);
  status[7]:=read_modem_status(RS232_address,3);
  if status[2]=1 then
     begin
        repeat
           write_transmit_buffer(RS232_address,output_byte);
        until keypressed;
        status[2]:=0;
     end;
  delay(50);
  serial_input_byte:=read_receive_buffer(RS232_address);
  setcolor(magenta);
  outtextxy(420,460,'Input serial data:'+Old_serial_input_string);
  setcolor(yellow);
  str(serial_input_byte,serial_input_string);
  outtextxy(420,460, 'Input serial data: '+serial_input_string);
  Old_serial_input_string:=serial_input_string;
  setbkcolor(cyan);
end;

Procedure Scan_keyboard;
begin
  key_pressed:=getkey;
  if key_pressed='LEFT' then led_selected:=led_selected-1;
  if key_pressed='RIGHT' then led_selected:=led_selected+1;
  if key_pressed=' ' then status[led_selected]:=1-status[led_selected];
  if (key_pressed='T') or (key_pressed='t') then
```

```
      begin
         setcolor(magenta);
         outtextxy(30,460,'Output serial data:'+serial_output_string);
         draw_message(100,290,100,60,cyan,'Input Serial data ', 0,1 yellow);
         gotoxy(22,20); readln(serial_output_string);
         draw_message(100,290,100,60,cyan,'Input serial data ',0,1,blue);
         setcolor(yellow);
         outtextxy(30,460,'Output serial data:'+serial_output_string);
      end;
   if (key_pressed='C') or (key_pressed='c') then
      begin
         closegraph;
         writeln('Configure RS232 port");
         write('Input baud rate (115200-9600-4800-2400-1200):"); readln(baud_rate_byte);
         write('Input parity (0=None, 1=Even, 3=Odd): '); readln(Parity_byte);
         write('Input data bit length (5, 6, 7, 8): '); readln(Data_length_byte);
         write('Input stop bit length (2, 1): '); readln(Stop_length_byte);
         write_data_format   (RS232_address, Baud_rate_byte, Parity_byte,
                              Data_length_byte, Stop_length_byte);
         initialize_graph;
         draw_panel;
         led_selected:=1;
      end;
end;

procedure draw_led_status;
var
   led_selected_old:integer;
begin
   led_selected_old:=led_selected;
   scan_keyboard;
   for i:=1 to 3 do draw_led(80+i*50,350,status[i]);
   output_input;

   for i:=4 to 7 do draw_led(180+i*50,350,status[i]);

   if led_selected>3 then led_selected:=3;
   if led_selected<1 then led_selected:=1;

   setlinestyle(0,0,3);
   setcolor(cyan);
   if led_selected_old<=3 then circle(80+50*led_selected_old,350,15);
   setcolor(yellow);
   if led_selected<=3 then circle(80+50*led_selected,350,15);
end;

(*Главная программа.*)
begin
   COM_address;
   initialize_graph;
   draw_panel;
   led_selected:=1;
   repeat
      draw_led_status;
   until (key_pressed='Q') or (key_pressed='q');
   closegraph;      (*Закрытие графического режима.*)
end.
```

Программа содержит четыре процедуры. Draw_panel используется для рисования виртуальной панели управления экспериментальной платой; Output_input – процедура ввода/вывода данных через последовательный порт. Процедуры и функции Write_modem_status(), write_transmit_buffer(), read_modem_port() и Read_receiver_buffer() записывают и считывают значения из регистров порта; они содержатся в библиотеке TPLIB1.PAS. Процедура scan_keyboard опрашивает клавиатуру, фиксируя нажатие шести функциональных клавиш: ←, →, клавиши пробела, **t**, **c** и **q**.

3.2.2. Описание программы RS232EXP

Программа выполняет следующие функции:

- сообщает о количестве установленных на компьютере последовательных портов;
- позволяет выбрать последовательный порт;
- настраивает формат последовательных данных;
- вводит байт и передает последовательные данные;
- изменяет состояние линий управления модемом (DTR и RTS);
- считывает последовательные данные;
- считывает состояние линий DSR, DCD и CTS.

Для запуска программы в среде Windows необходимо нажать на кнопку **Пуск**, выбрать в меню пункт **Выполнить** и ввести ее полное имя: путь + RS232EXP, после чего щелкнуть по кнопке **OK**. На экране появится сообщение о количестве установленных COM-портов и их базовых адресах (рис. 3.6). Нажмите **OK**. Вы увидите окно, изображенное на рис. 3.7.

Здесь требуется ввести номер порта (1–4), к которому подсоединена экспериментальная плата. Сделав это, щелкните по кнопке **OK**. Появится окно управления экспериментальной платой последовательного порта (рис. 3.8).

Кнопки **0-1** позволяют изменять состояние линий DTR и RTS, причем высокий уровень индицируется красным светодиодом. Вы можете ввести данные, которые необходимо передать в поле данных панели. При нажатии на кнопку **0-1** эти данные будут постоянно передаваться, пока кнопка не будет нажата еще раз. С помощью кнопки **Get it** вы обновляете состояние линий DSR, DCD и CTS. Принятые данные также отображаются на экране. Кнопка **Change port** дает возможность выбрать другой порт, переопределить скорость, режим проверки, длину блока данных и количество стоповых битов. Внизу окна имеется информационная панель.

Соединив приемную и передающую линии, допустимо провести тестирование порта. В этом случае значения передаваемых и принимаемых данных должны совпадать. Когда программа передает данные, вид сигнала в линии можно наблюдать с помощью осциллографа. Разрешается передавать различные данные, изменять скорость, режим проверки и длину блока данных – это позволяет глубже понять процессы, происходящие при передаче последовательных данных.

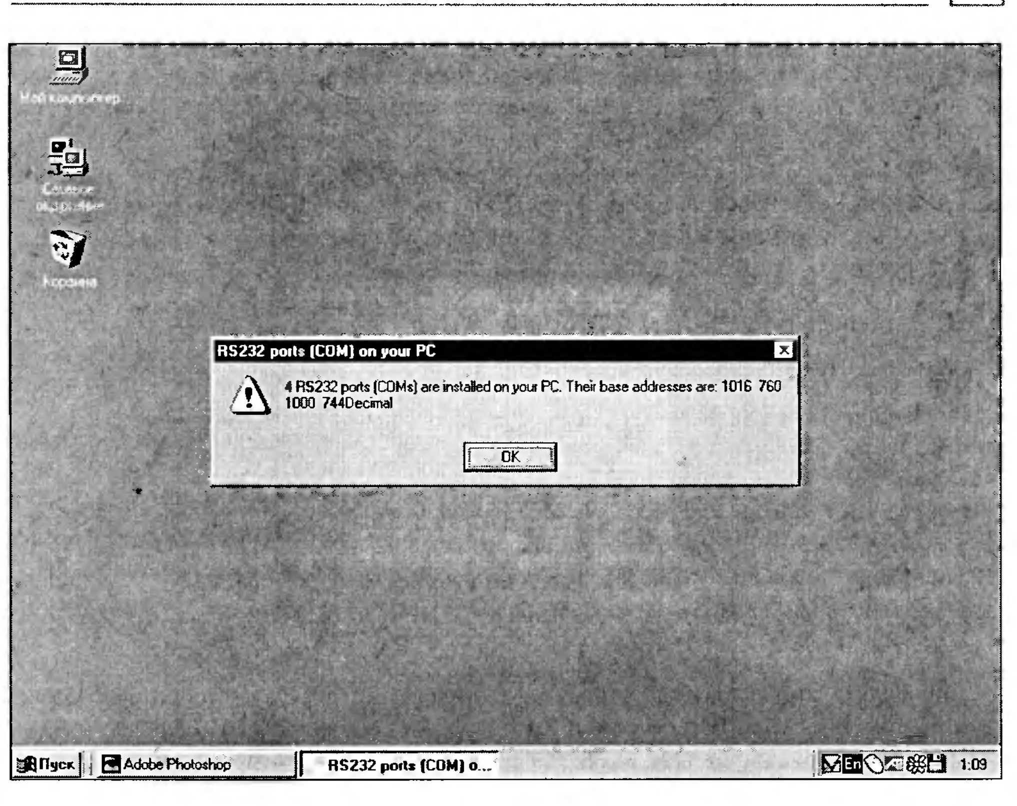


Рис. 3.6. Сообщение, выдаваемое программой RS232EXP

Текст программы RS232EXP

```
  Объявление функции библиотеки динамической компоновки WLIB1 DLL
  Объявляемые функции   RS232()  Bit_weight() Write_interrupt_enable()
                        Read_interrupt_identification()
                        write_data_format()  write transmit buffer()  write_modem_status()
                        write_receive_buffer() и read_modem_status()
Declare Function RS232 Lib  C \IOEXP\WLIB1 dll  (ByVal X As Integer) As Integer
Declare Function Bit_weight Lib  C \IOEXP\WLIB1 dll  (ByVal X As Integer) As Integer
Declare Function Write_interrupt_enable Lib  C \IOEXP\WLIB1 dll  (ByVal address As Integer
ByVal datax As Integer) As Integer
Declare Function Read_interrupt_identification Lib  C \IOEXP\WLIB1 dll  (ByVal address As
Integer) As Integer
Declare Function write_data_format Lib  C \IOEXP\WLIB1 dll  (ByVal address As Integer  ByVal
Baud As Integer  ByVal parity As Integer  ByVal Data_byt As Integer  ByVal Stop_bit As
Integer) As Integer
Declare Function Write_transmit_buffer Lib  C \IOEXP\WLIB1 dll  (ByVal address As Integer
ByVal datax As Integer) As Integer
Declare Function Write_modem_status Lib  C \IOEXP\WLIB1 dll  (ByVal address As Integer  ByVal
RTS As Integer  ByVal DTR As Integer) As Integer
```

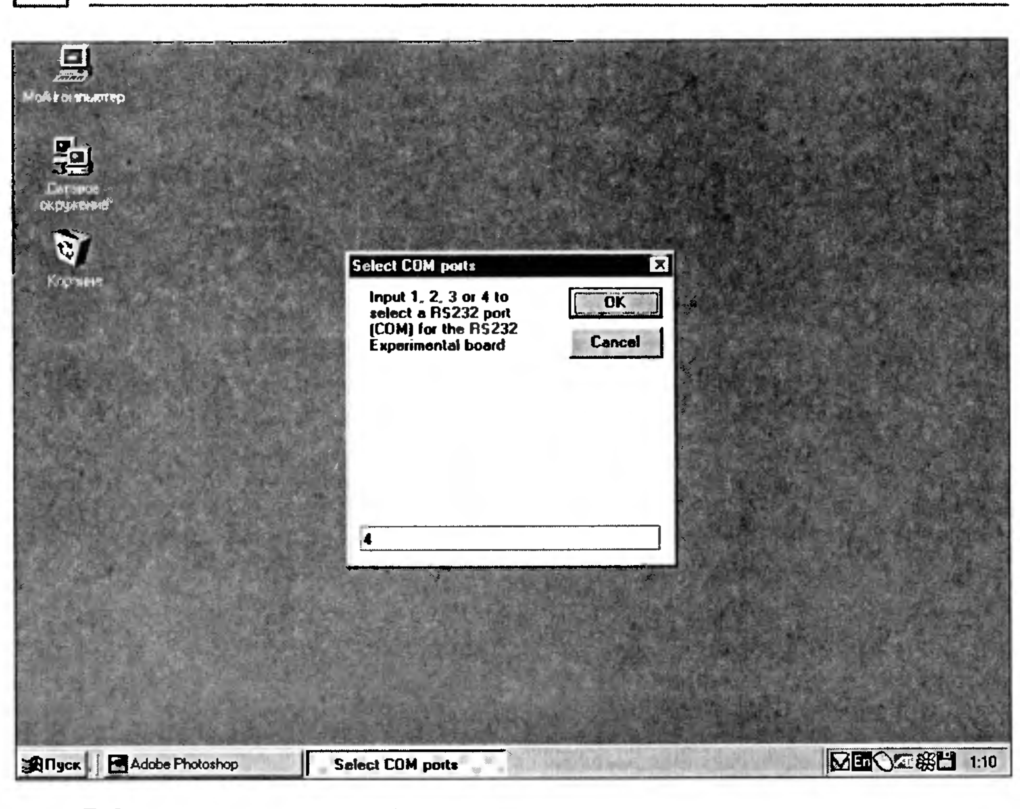


Рис. 3.7. Окна выбора используемого СОМ-порта

```
Declare Function Read_receive_buffer Lib "C:\IOEXP\WLIB1 dll" (ByVal address As Integer) As
Integer
Declare Function Read_modem_status Lib"C.\IOEXP\WLIB1.dll" (ByVal address As Integer, ByVal X
As Integer) As Integer

Sub Command1_click()
   status(4)=Read_modem_status(RS232_address,2)
   status(5)=Read_modem_status(RS232_address,1)
   status(6)=Read_modem_status(RS232_address,3)
   For iq=4 to 6
   If status(iq)=1 Then Shape1(iq).BackColor=&HFF& Else Shape1(iq).BackColor=black
   Next iq

   Label3.Caption=Format$(Read_receive_buffer(RS232_address))
End Sub

Sub command1_Mouse_move(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label7 Caption="Get the status and read the serial data'
End Sub

Sub Command2_Click(Index As Integer)
   'Изменение состояния выходных линий.
   status(index)=1-status(index)
   If status(index)=1 Then Shape1(index) BackColor=&HFF& Else Shape1(index) BackColor=black
```

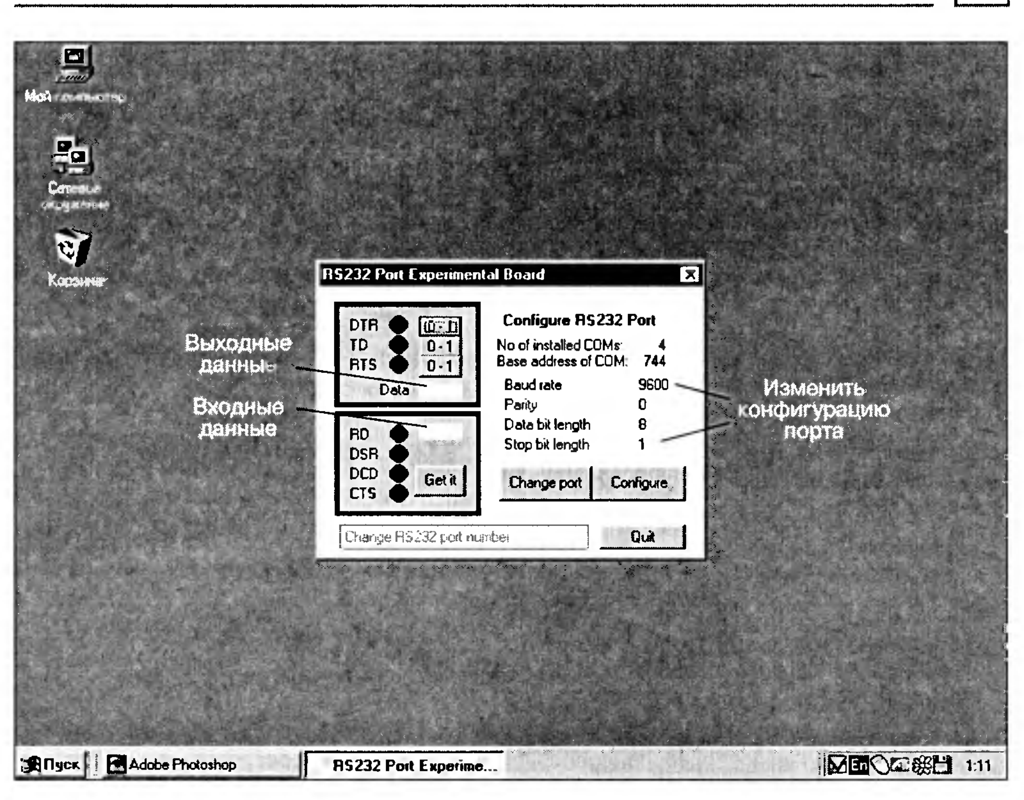


Рис. 3.8. Окно управления экспериментальной платой последовательного порта

```
 Ввод данных в регистр состояния модема
dummy=Write_modem_status(RS232_address, status(2), status(0))

 Обновление входных линий
status(4)=Read_modem_status(RS232_address,2)
status(5)=Read_modem_status(RS232_address,1)
status(6)=Read_modem_status(RS232_address,3)
For iq=4 to 6
If status(iq)=1 Then Shape1(iq) BackColor=&HFF& Else Shape1(iq) BackColor=black
Next iq

 Считывание и отображение входных данных
Label3 Caption=Format$(Read_receive_buffer(RS232_address))

 Вывод последовательных данных
If status(1)=1 Then
    Do
    dummy=Write_transmit_buffer(RS232_address,Val(text1 Text))
    DoEvents
    Loop While status(1)=1
End If
End Sub
```

```
Sub Command2_MouseMove(index As Integer, Button As Integer, Shift As Integer, X As Single,
Y As Single)
   'Отображение справочной информации.
   Label7.Caption="Change the status of the output line"
End Sub

Sub Command3_Click()
   'Настройка выбранного последовательного порта.
   baud_rate=Val(text2(0).Text)              'Установка скорости.
   parity=Val(text2(1).Text)                 'Установка проверки на четность.
   data_bit_length=Val(text2(2).Text)        'Установка длины блока данных.
   stop_bit_length=Val(text2(3) Text)        'Установка длины стопового бита.
   dummy=write_data_format(RS232_address,baud_rate,parity,data_bit_length,  stop_bit_length)
'Запись конфигурации в регистр формата данных.
End Sub

Sub Command3_Mouse_move(Button As Integer,Shift As Integer,X As Single,Y As Single)
   Label7.Caption="Change the configuration of RS232 port"
End Sub

Sub Command4_Click()
   End
End Sub

Sub Command4_Mouse_move(Button As Integer,Shift As Integer,X As Single,Y As Single)
Label7.Caption="Quit the program"
End Sub

Sub Command5_Click()
   'Выбор другого порта.
   dummy=MsgBox(Str(RS232(0)) & "RS232 ports (COMs) are installed on your PC  Their base addresses
are.' & Format$(RS232(1), "###") & " " & Format$(RS232(2), "###") & " " & Format$(RS232(3),
"###") & " " & Format$(RS232(4), "###") & "Decimal", 48, "RS232 ports (COM) on your PC")
'Отображение информации о порте.
   RS232_number=Val(InputBox$("Input 1, 2, 3 or 4 to select a RS232 port (COM) for the
   Mini-Lab Data Logger/Controller", "Select COM ports"))              'Выбор COM-порта.
   RS232_address=RS232(RS232_number)         'Получение базового адреса выбранного порта.
   Label2.Caption="Selected COM port: " & Format(RS232_number)        'Отображение информации
                                                                      'о выбранном порте.
   Label4.Caption="Base address of COM· " & Format(RS232_address)
End Sub

Sub Command5_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label7.Caption="Change RS232 port number'
End Sub

Sub Form_Load()
   For i=0 To 11
      status(i)=0
   Next i

   dummy= MsgBox(Str(RS232(0)) & 'RS232 ports (COMs) are installed on your PC. Their base
addresses are:" & Format$(RS232(1), "###') & " " & Format$(RS232(2), "###") & " " &
Format$(RS232(3), "###') & " " & Format$(RS232(4), "###") & "Decimal", 48, "RS232 ports (COM)
on your PC")
   RS232_number=Val(InputBox$("Input 1, 2, 3 or 4 to select a RS232 port (COM) for the RS232
Experimental board", "Select COM ports'))
```

```
    RS232_address=RS232(RS232_number)
    Label2 Caption= No of installed COMs    & Format(RS232_number)
    Label4 Caption= Base address of COM    & Format(RS232_address)
    baud_rate=9600
    parity=0
    data_bit_length=8
    stop_bit_length=1
    text2(0) Text=Format$(baud_rate)
    text2(1) Text=Format$(parity)
    text2(2) Text=Format$(data_bit_length)
    text2(3) Text=Format$(stop_bit_length)

    dummy=write_data_format(RS232_address  baud_rate  parity data_bit_length  stop_bit_length)

End Sub

Sub Label3_MouseMove(Button As Integer Shift As Integer X As Single Y As Single)
     Отображение принятых данных
    Label7 Caption= Value of the serial input data
End Sub

Sub Label6_MouseMove(index As Integer Button As Integer Shift As Integer X As Single Y As
Single)
    Select Case index
    Case 0
        Label7 Caption= Baud rate=115200  19200  9600  2400  etc
    Case 1
        Label7 Caption= 0=No Parity  1=Odd Parity  3=Even Parity
    Case 2
        Label7 Caption= Input 5  6  7 or 8 to select the data bit length
    Case 3
        Label7 Caption= Input 1 or 2 to select Stop bit
    End Select
End Sub

Sub Text1_MouseMove(Button As Integer Shift As Integer X As Single Y As Single)
     Ввод данных для передачи
    Label7 Caption= Input the serial data to be sent out
End Sub

Sub Text2_Change(index As Integer)
    Select Case index
        Case 0
        Case 1
        Case 2
        Case 3
    End Select
End Sub
```

3.3. Программное обеспечение для экспериментальной платы игрового порта

В разделе представлены тексты программ управления экспериментальной платой игрового порта и необходимые пояснения к ним

3.3.1. Описание программы GAMEEXP.PAS

Программа выполняет следующие функции:

- считывает состояния двух (или четырех) цифровых входов;
- измеряет длительность периода одновибратора для двух (четырех) аналоговых каналов;
- калибрует аналоговые каналы;
- измеряет сопротивление резистора.

После запуска программы на экране появляется виртуальная панель управления (рис. 3.9), где изображены четыре виртуальных светодиода и восемь виртуальных контактов, соответствующих светодиодам и контактам на экспериментальной плате игрового порта.

Внизу расположена панель помощи, на которой показаны клавиши управления программой с кратким описанием их действий:

- **[ARROW]** (клавиши управления курсором) – выбор аналогового канала;
- **[SPASE]** (пробел) – считывание сопротивления выбранного канала и состояния входов;
- **[C]** или **[c]** – калибровка выбранного аналогового канала;
- **[Q]** или **[q]** – выход из программы.

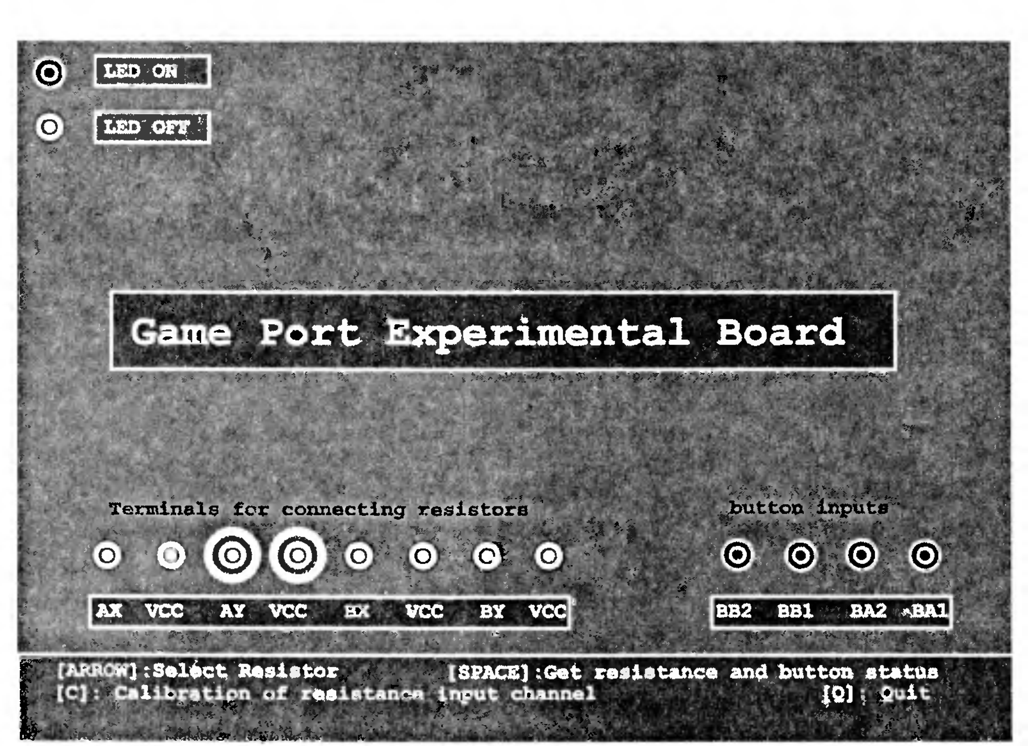


Рис. 3.9. Виртуальная панель управления экспериментальной платой игрового порта

Посредством клавиш ← и → можно указать входной аналоговый канал. Выбранный канал выделяется двумя окружностями вокруг виртуального контакта на панели. Нажав на клавишу пробела при наличии резистора, соединенного с входом, вы получите длительность периода одновибратора. Если этот канал откалиброван, выводится также сопротивление резистора. Перед калибровкой (она осуществляется с помощью клавиши **C**) необходимо замкнуть накоротко входные контакты, а затем соединить с контактом резистор с известным сопротивлением. При работе программы данные считываются и с цифровых входов порта. Состояние битов данных индицируется виртуальными светодиодами.

Текст программы GAMEEXP.PAS

```
Program Game_tester;
(*Программа управления экспериментальной платой игрового порта.*)
uses
  graph,crt,dos;
const
  game_address=513;
var
  i,led_selected,Time1,Time2:integer;
  ch:char;
  status:array[1..18] of integer,
  period:real;
  interval_0,Interval_R,standard_R:array[1..4] of real,
  key_pressed:string[10];
  strx·string,

(*Загрузка файлов библиотек.*)
{$I c:\ioexp\tplib2.pas}
{$I c:\ioexp\tplib1.pas}

Function Resistance(period.real; led_selected:integer):real;
(*Вычисление сопротивления при известном периоде.*)
(*Необходима калибровка.*)
var
  dummy,dummy2.real;
begin
  dummy2:=interval_R[led_selected]-interval_0[led_selected];
  if abs(dummy2)<0.0001 then dummy2:=1;
  dummy:=standard_R[led_selected]/dummy2*(period-interval_0[led_selected]);
  if abs(dummy)>200 then resistance:=200 else resistance:=dummy;
end,

procedure draw_panel;
(*Рисование панели управления.*)
begin
  for i.=1 to 16 do status[i]·=0;
  setbkcolor(cyan),
  for i:=1 to 8 do draw_led(20+i*40,350,status[i]);
  for i:=1 to 4 do draw_led(420+i*40,350,status[i+8]);
  draw_led(20,20,1);
  draw_message(50,20,70,20,lightblue,'LED  ON',0,1,yellow);
  draw_led(20,60,0);
  draw_message(50,60,70,20,lightblue,'LED  OFF',0,1,yellow);
```

```
   setfillstyle(1,magenta);
   bar(1,420,800,480);
   settextstyle(0,0,1);
   outtextxy(10,425,'[ARROW]:Select  Resistor    [Space]:Get resistance and button status');
   outtextxy(10,425,'[C]:Calibration of resistance input channel[Q]:Quit');
   draw_message(60,200,500,50,blue,'Game  port Experimental  board',0,2,yellow);
   draw_message(50,390,305,20,blue,'AX       VCC AY       VCC BX       VCC BY
              VCC',0,1,yellow);
   draw_message(450,390,149,20,blue,'BB2         BB1 BA2 BA1',0,1,yellow);
   draw_message(55,320,280,20,cyan,'Terminals for connecting resistors',0,1,red);
   draw_message(450,320,120,20,cyan,'Button  inputs',0,1,red);
end;
procedure output_input;
var
   output_byte,input_byte:byte;
begin
   (*Считывание состояния кнопок.*)
   status[9]:=read_game_port(8);
   status[10]:=read_game_port(7);
   status[11]:=read_game_port(6);
   status[12]:=read_game_port(5);
end;
procedure scan_keyboard;
begin
   key_pressed:=getkey;
   if key_pressed='LEFT' then led_selected:=led_selected-1;
   if key_pressed='RIGHT' then led_selected:=led_selected+1;
   if key_pressed=' 'then
      begin
         period:=interval_game_port(led_selected);
         str(period*0.838:10:2,strx);
         draw_message(25,460,250,10,magenta,'Interval (us):  '+strx,0,1, lightcyan);
         str(resistance(period,led_selected):10:2,strx);
         draw_message(370,460,250,10,magenta,'Resistance (kOhm):  '+strx,0,1, lightcyan);
      end;
   if (key_pressed='Q') or (key_pressed='q') then
      begin
         closegraph;
         clrscr;
         writeln('Calibration of resistance input channels CH',led_selected:4);
         writeln('Short the input terminal for the selected channel');
         writeln('Press RETURN to continue');
         readln;
         interval_0[led_selected]:=interval_game_port(led_selected);
         writeln('Connect the standard resistor to the selected channel');
         write('Input the resistance of the resistor');
         readln(standard_R[led_selected]);
         interval_R[led_selected]:=interval_game_port(led_selected);
         initialize_graph;
         draw_panel;
      end;
end;
```

```
procedure Draw_led_status,
var
  led_selected_old integer,
begin
  led_selected_old =led_selected,
  scan_keyboard,
  output_input
  for i =1 to 4 do draw_led(420+i*40,350,status[i+8]),

  if led_selected>4 then led_selected =4,
  if led_selected<1 then led_selected =1,

  setlinestyle(0,0,3),
  setcolor(cyan),
  if led_selected_old<=4 then
     begin
        circle(-20,+80*led_selected_old 350,15),
        circle(-20,+80*(led_selected_old+40),350,15),
     end,
  setcolor(yellow)
  if led_selected<=4 then
     begin
        circle(-20,+80*led_selected,350,15),
        circle(-20,+80*(led_selected)+40),350,15),
     end,
end,

(*Главная программа *)
begin
  initialize_graph,
  draw_panel,
  init_8253,
  led_selected =1,
  repeat
     draw_led_status
  until (key_pressed= Q ) or (key_pressed= q ),
  closegraph,     (*Закрытие графического режима *)
end
```

Программа содержит четыре процедуры и одну функцию; кроме того, применяются процедуры и функции из программных библиотек ресурсов. `Draw_panel` рисует виртуальную панель управления. Процедуры `Draw_led()` и `Draw_message()`, содержащиеся в библиотеке TPLIB2.PAS, выводят на экран монитора изображения виртуальных светодиодов и поясняющие надписи. Процедура `Output_input` организует ввод/вывод данных через игровой порт. `Scan_keyboard` сканирует клавиатуру и фиксирует нажатие клавиш →, ←, клавиши пробела, **c** или **q**. Она использует функцию `getkey` из библиотеки TPLIB2.PAS. Функции `Read_game_port()` и `Interval_game_port`, находящиеся в библиотеке TPLIB1.PAS, считывают состояние цифровых входов порта и измеряют длительность периода одновибратора соответственно. Функция `Resistance()` вычисляет значение сопротивления на основании параметров калибровки.

3.3.2. Описание программы GAMEEXP

Эта программа выполняет следующие функции:

- считывает состояния двух (или четырех) цифровых входов;
- измеряет длительность период одновибратора для двух (четырех) аналоговых каналов;
- калибрует аналоговые каналы;
- измеряет сопротивление резистора.

Для запуска программы необходимо нажать кнопку **Пуск** и выбрать в меню пункт **Выполнить.** Затем следует ввести ее полное имя: путь + GAMEEXP. После нажатия кнопки **OK** на экране появится окно управления экспериментальной платой игрового порта (рис. 3.10). Программа выполняет функции, аналогичные DOS-версии.

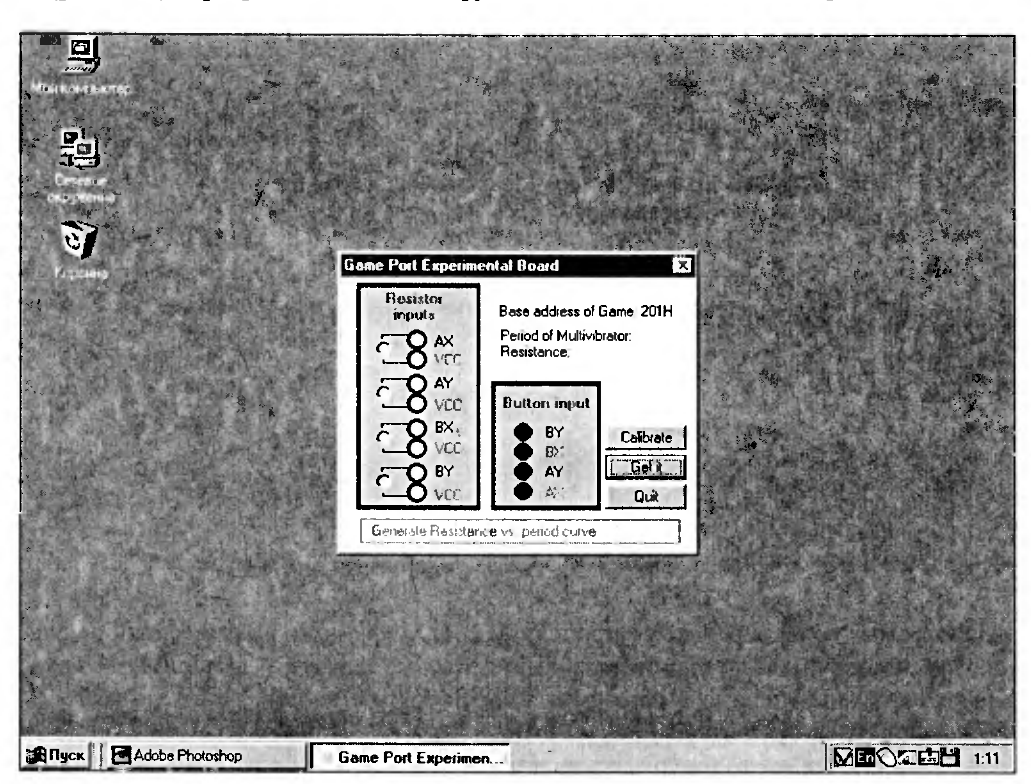


Рис. 3.10. Окно управления экспериментальной платой игрового порта

Текст программы GAMEEXP

```
Declare Function Bit_weight Lib  C \IOEXP\Wlib dll' (ByVal X As Integer) As Integer
Declare Function Read_game_port Lib  C \IOEXP\Wlib.dll" (ByVal X As Integer) As Integer
Declare Function Write_game_port Lib  C \IOEXP\Wlib dll' () As Integer
Declare Function Interval_game_port Lib  C \IOEXP\Wlib dll' (ByVal led_selected As Integer)
As Integer
```

```
Sub Command1_click()
   DoEvents
   For i=12 to 15
   status(i)=read_game_port(20-i)
If status(i)=1 Then Shape1(i) BackColor=$HFF$ Else Shape1(i) BackColor=black
   Next i
   DoEvents
   period=Interval_game_port(led_selected)* 838
   If period<>0 Then
   Label2 Caption= Period of multivibrator   & Format(period, ##### # )
   Else
   Label2 Caption= Error reading multivibrator
   End If

   Label5 Caption= Resistance   & Format(resistance(period)  ### # )
End Sub

Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Label6 Caption= Get button status and period of multivibrators
End Sub

Sub Command2_click(index As Integer)

   status(index)=1-status(index)
   If status(index)=1 Then Shape1(index) BackColor=$HFF$ Else Shape1(index) BackColor=black

   For i=12 to 15
   status(i)=(read_game_port(i-11) and Bit_weight(i-11))/Bit_weight(i-11)
   If status(i)=1 Then Shape1(i) BackColor=$HFF$ Else Shape1(i) BackColor=black
   Next i
End Sub

Sub Command3_click()
   MsgBox( Short the terminals of the selected resistance input  The selected channel is
        & led_selected)
   interval_0(led_selected)=Interval_game_port(led_selected)* 838
   MsgBox( Connect a standard resistor to the selected resistance input  The selected
        channel is   & led_selected)
   Standard_r(led_selected)=InputBox( Input the resistance   ,  Calibration of resistance
   input ,  1 )
   interval_R(led_selected)=Interval_game_port(led_selected)* 838
End Sub

Sub Command1_MouseMove(Button As Integer, Shift As Integer  X As Single  Y As Single)
   Label6 Caption= Generate resistance vs  period curve
End Sub

Sub Command4_Click()
   End
End Sub

Sub Command4_MouseDown(Button As Integer  Shift As Integer, X As Single, Y As Single)
   Label6 Caption= Quit the program
End Sub

Sub Form_load()
   For i=0 to 16
       status(i)=0
```

```
  Next i

End Sub

Sub Option1_Click(index As Integer)
  led_selected=index+1
End Sub

Sub Option1_MouseMove  (index As Integer, Button As Integer, Shift As Integer, X As Single, Y
                        As Single)
  Label6.Caption="Select No." & index+1 & "resistance input channel"
End Sub

Function resistance(ByVal period As Single) As Single
  dummy2=(interval_R(led_selected)-interval_0(led_selected))
  If Abs(dummy2)<.001 Then dummy2=1
  dummy=Standard_r(led_selected)/dummy2*(period-interval_0(led_selected))
  If Abs(dummy)>200 Then resistance=200 Else resistance=dummy
End Function
```

3.4. Программные библиотеки ресурсов

Ниже приведены тексты библиотек ресурсов TP6 (TPLIB1.PAS и TPLIB2.PAS для DOS) и библиотеки динамической компоновки для Windows (WLIB.PAS). Подробные комментарии к процедурам и функциям даны в тексте.

Библиотека ресурсов TPLIB1.PAS

```
(*Библиотека ресурсов TP6 № 1.*)
(*Название библиотеки: TPLIB1.PAS.*)
(*Процедуры для управления параллельным, последовательным и игровым портами *)
(*Эта библиотека может быть включена в пользовательскую программу.*)

var
  P_address,RS232_address:integer;

(*-Библиотека ресурсов № A1 (определение базового адреса LPT) -*)
Procedure Centronic_Address
(* $000:$0408 содержит базовый адрес для LPT1
  $000:$040A содержит базовый адрес для LPT2
  $000:$040C содержит базовый адрес для LPT3
  $000:$040B содержит базовый адрес для LPT4
  $000:$0411 содержит количество параллельных портов *)
var
  lpt:array[1..4] of integer;
  number_of_lpt,LPT_number,code:integer;
  kbchar:char;
begin
     clrscr;
     LPT_number:=1;                     (*Для установки принтера по умолчанию *)
     number_of_lpt:=mem($0000:$0411);   (*Считывание количества установленных параллельных
                                        портов *)
     number_of_lpt:=(number_of_lpt and (128+64)) shr 6,      (*Битовые преобразования.*)
     lpt[1]:=memw($0000:$0408);         (*Процедура считывания из памяти.*)
     lpt[2]:=memw($0000:$040A);
     lpt[3]:=memw($0000:$040C);
```

```
      lpt[4] =memw($0000 $040E)
      textbackground(blue)  clrscr,
      textcolor(yellow)  textbackground(red), window(10,22,70,24), clrscr,
      writeln( Number of LPT installed   number_of_lpt 2),
      writeln( Addresses for LPT1 to LPT4   , lpt[1] 3,   ,lpt[2] 3,    ,lpt[3] 3    lpt[4] 3),
      write( Select LPT to be used(1,2,3,4)   ),
      delay(1000),
      if number_of_lpt>1 then  begin {Выбор порта, если установлено несколько портов }
                                    repeat
                                      kbchar =readkey,            (*Считывание значения
                                                                  с вводимой клавиши *)
                                      val(kbchar,LPT_number,code), (*Преобразование символа
                                                                  в число *)
                                      until (LPT_number>=1) and   (LPT_number<=4) and
                                                                  (lpt[LPT_number]<>0),
                                 end,
      clrscr
      P_address =lpt[LPT_Number],
      writeln( Your selected printer interface  LPT",LPT_number 1),
      write( LPT address   ,RS232_address 3),
      delay(1000)
      textbackground(black)  window(1 1 80,25), clrscr,
end,

(*-Библиотека ресурсов № A2 (нахождение веса бита) -*)
Function  bit_weight(bit byte) byte,
var
  i,dummy integer
begin
  if bit=1 then bit_weight =1
  else
      begin
          dummy =1
          for i =1 to bit-1 do dummy =2*dummy,
          if dummy=0 then dummy =1,
          bit_weight =dummy,
      end
end

(*-Библиотека ресурсов № A3 (считывание данных в компьютер) -*)
Function  Read_status_port(P_address integer) byte,
var
  byte1 byte,
begin
  byte1 =port(P_address+1),                (*Считывание байта из регистра состояния *)
  byte1 =byte1 and 120,   (*01111000 (от старшего к младшему) and 0dddd  = 0dddd000 *)
  Read_status_port =byte1 shr 3,           (*Сдвиг на 3 бита вправо,
                                           Read_status_port=0000hhhh *)
end,

(*-Библиотека ресурсов № A4 (запись данных в регистр данных компьютера) -*)
Procedure Write_data_port(P_address integer, port_data byte),
(*Линии порта данных не инвертированы *)
```

```
begin
  port(P_address):=port_data;      (*Ввод байта в регистр данных.*)
end;

(*-Библиотека ресурсов № A5 (запись данных в регистр управления).-*)
Procedure Write_control_port(P_address:integer; port_data:byte);
(*Биты 0, 1 и 3 инвертированы. Требуются преобразования над битами.*)
begin
  if port_data and 1=1 then port_data:=port_data and (255-1)
     else port_data:=port_data or 1;
  if port_data and 2=2 then port_data:=port_data and (255-2)
     else port_data:=port_data or 2;
  if port_data and 8=8 then port_data:=port_data and (255-8)
     else port_data:=port_data or 8;
  port(P_address+2):=port_data;    (*Ввод байта данных в регистр управления.*)
end;

(*-Библиотека ресурсов № A6 (определение базовых адресов COM-портов).-*)
Procedure COM_address;
(* $0000:$0400 содержит базовый адрес порта COM1,
  $0000:$0402 содержит базовый адрес порта COM2,
  $0000:$0404 содержит базовый адрес порта COM3,
  $0000:$0406 содержит базовый адрес порта COM4,
  $0000:$0411 содержит количество COM-портов.*)
var
  COM:array[1..4] of integer;
  COM_number,number_of_COM,code:integer;
  Kbchar:char;
begin
  clrscr;
  COM_number:=1;                    (*Установка порта по умолчанию.*)
  Number_of_COM:=mem($0000:$0411);  (*Считывание количества COM-портов.*)
  Number_of_COM:=(Number_of_COM and (8+4+2)) shr 1;
  COM[1]:=memw($0000:$0400);        (*Процедура считывания из памяти.*)
  COM[2]:=memw($0000:$0402);
  COM[3]:=memw($0000:$0404);
  COM[4]:=memw($0000:$0406);
  textbackground(blue); clrscr;
  textcolor(yellow); Textbackground(red); window(10,22,70,24); clrscr;
  writeln('Number of COM installed:', Number_of_COM:2);
  writeln('Addresses for COM1 to COM4:', COM[1]:3, COM[2]:3, COM[3]:3, COM[4]:3);
  write('Select COM to be used (1,2,3,4):');
  delay(1000);
  if number_of_COM>1 then begin (*Выбор порта, если установлено несколько портов.*)
     repeat
        kbchar:=readkey;            (*Считывание значения с вводимой клавиши.*)
        val(kbchar,COM_number,code); (*Преобразование символа в число.*)
     until (COM_number>=1) and (COM_number<=4) and (COM[COM_number]<>0);
end;
  clrscr;
  RS232_address:=COM[COM_number];
  writeln('Your selected RS232 interface: COM', COM_number:1);
  write('RS232 address:', RS232_address:4);
```

```
  delay(1000);
  Textbackground(black); window(1,1,80,25); clrscr;
end;

(*-Библиотека ресурсов № A7 (запись в регистр разрешения прерываний).-*)
Procedure write_interrupt_enable(RS232_address, output_byte:integer);
begin
  port(RS232_address+1):=Output_byte;
end;

(*-Библиотека ресурсов № A8 (чтение из регистра идентификации прерываний).-*)
Function  read_interrupt_identification(RS232_address:integer):integer;
begin
  read_interrupt_identification:=port(RS2132_address+2);
end;

(*-Библиотека ресурсов № A9 (запись данных в регистр формата данных).-*)
Procedure Write_data_format (RS232_address, Baud, Parity, Data_bit, Stop_bit:integer);
var
  byte1,byte2,output_byte:byte;
  divisor:integer;
begin
  divisor:=115200 div Baud;
  if divisor<=255 then begin byte1:=divisor; byte2:=0; end;
  if divisor>255 then begin byte1:=divisor mod 256; byte2:=divisor div 256;
                   end;
  output_byte:=(data_bit-5)+4*(stop_bit-1)+8*parity;
  port(RS232_address+3):=128;          {Загрузка формата последовательных данных, первый бит
                                        регистра равен 1.}
  port(RS232_address+0):=byte1;        {Младший байт делителя равен 1.}
  port(RS232_address+1):=byte2;        {Старший байт делителя равен 0.}
  port(RS232_address+3):=output_byte;  {Загрузка делителя и других параметров.}
end;

(*-Библиотека ресурсов № A10 (запись данных в буферный регистр передатчика).-*)
Procedure  write_transmit_buffer(RS232_address,output_byte:integer);
begin
  port(RS232_address):=output_byte;
end;

(*-Библиотека ресурсов № A11 (запись данных в регистр состояния модема).-*)
procedure write_modem_status(RS232_address, RTS, DTR:integer);
(*RTS и DTR или 1, или 0. RTS и DTR инвертируются с помощью MAX238
на экспериментальной плате.*)
(*RTS=бит 1, DTR=бит 0 регистра управления модемом, смещение 04h.*)
begin
  RTS:=1-RTS;
  DTR:=1-DTR;
  port(RS232_address+4):=RTS*2+DTR;        (*Запись в регистр 04h.*)
end;

(*-Библиотека ресурсов № A12 (чтение данных из буферного регистра приемника).-*)
Function read_receive_buffer(RS232_address:integer):integer;
```

```
Begin
  read_receive_buffer:=port(RS232_address);
end;

(*-Библиотека ресурсов № A13 (чтение данных из регистра состояния модема).-*)
Function read_modem_status(RS232_address, x:integer):integer;
(*x=1 - выбор бита DCD, x=2 - выбор бита DSR, x=3 - выбор бита CTS.*)
(*DCD=бит 7, DSR=бит 5, CTS=бит 4 регистра состояния модема, смещение 06h.*)
(*Все биты инвертируются с помощью MAX238 на экспериментальной плате.*)
var
  input_byte:byte;
begin
  input_byte:=port(RS232_address+6);
  case x of
     1:  Read_modem_status:=1-round((input_byte and 128)/128);
     2:  Read_modem_status:=1-round((input_byte and 32)/32);
     3:  Read_modem_status:=1-round((input_byte and 16)/16);
  end;
end;

(*-Библиотека ресурсов № A14 (чтение регистра игрового порта).-*)
Function read_game_port(bitx:integer):integer;
(*Адрес игрового порта: 201h.
Bitx (1-8) выбирает состояние AX, AY, BX, BY, BA1, BA2, BB1 и BB2.*)
var
  input_byte:byte;
begin
  input_byte:=port($201);
read_game_port:=round((input_byte and bit_weight(bitx))/bit_weight(bitx));
end;

(*-Библиотека ресурсов № A15 (запись в регистр игрового порта).-*)
Function write_game_port;
(*Вводит 0 в игровой порт для запуска мультивибратора.*)
begin
  port($201):=0;
end;

(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
  time1,time2,dummy:integer;

Procedure init_8253;
(*Настройка 8253.*)
begin
(*Управляющее слово = b6h = 10110110b:
  10 = выбор счетчика 2;
  11 = сначала чтение/запись младшего байта, затем старшего;
  011 = режим 3;
  0 = шестнадцатибитовый двоичный счет.*)
  port($43):=$b6;             (*Запись слова в управляющий регистр 8253.*)
  port($42):=255;             (*Загрузка младшего байта.*)
  port($43):=255;             (*Загрузка старшего байта.*)
```

```
  port($61) =port($61) or 1   (*Выключение динамика *)
  port($43) =$80              (*80h - команда фиксации для счетчика 3 *)
end,

Function read_8253 integer,
(*Считывание младшего и старшего байтов счетчика *)
var
  low_byte high_byte byte,
begin
  low_byte =port($42),
  high_byte =port($43),
  read_8253 =low_byte+256*high_byte,
end,

var
  i integer,
begin
  init_8253,
  for i =1 to 100 do i =i,
  i =0,
  dummy =bit_weight(x),
  port($201) =0,
  time1 =read_8253,
  repeat i =i+1 until (port($201) and dummy=0) or (i>=5000),
  time2 =read_8253,
  interval_game_port =time2-time1,
  if i>=5000 then interval_game_port =0,
end,
```

Библиотека ресурсов TPLIB2.PAS

```
(*Библиотека ресурсов TP6 № 2 *)
(*Название библиотеки  TPLIB2 PAS *)
(*Процедуры для управления графикой и чтения клавиатуры *)
(*Эта библиотека может быть включена в пользовательскую программу *)

(*-Библиотека ресурсов № 01 (инициализация графического режима) -*)
Procedure initialize_graph,
var
  gd,gm integer,
  radius integer,
begin
  gd =detect, initgraph(gd,gm,    ),
end,

(*-Библиотека ресурсов № 02 (рисование светодиода) -*)
procedure draw_led(x,y integer, status byte)
(*X и Y - координаты центра  Status - включено или выключено *)
begin
  setcolor(red)
  setlinestyle(1 1 2),
  if status=1 then setfillstyle(1,red)
     else
         begin
             setcolor(white),
             setfillstyle(1,white)
         end,
```

```
  pieslice(x,y,0.360,10);
  setcolor(magenta),
  circle(x,y,10);
  setcolor(yellow);
  circle(x,y,5);
end;

(*-Библиотека ресурсов № 03 (рисование окон и вывод сообщений).-*)
procedure draw_message(x1,y1,width,height,color_box:integer, message.string;
font,size_text,color_text integer);
(* x1,y1 - координаты левого края надписи;
  width и height - ширина и высота окна сообщения;
  color_box - цвет окна сообщения;
  message - сообщение, выводимое на экран;
  Font,size_text,color_text - шрифт, размер и цвет текста.*)
begin
  setfillstyle(1,color_box);
  bar(x1,y1-round(height/2),x1+width,y1+round(height/2));
  setcolor(color_text);
  settextstyle(font,0,size_text);
  outtextxy(x1+5,y1-4,message);
end;

(*-Библиотека ресурсов № 04 (считывание символов при нажатии функциональных клавиш).-*)
function getkey:string;
var
  ch:char;
begin
ch:=readkey;   (*Считывание символа с клавиатуры.*)
if ch=#0 then  (*Если нажата дополнительная клавиша, то выполняются следующие процедуры.*)
  begin
      ch:=readkey; (*Считывание клавиши снова для получения кода символа.*)
      if ch=#72 then getkey='UP';           (*Стрелка вверх = #72.*)
      if ch=#80 then getkey='DOWN';         (*Стрелка вниз = #80.*)
      if ch=#75 then getkey='LEFT';         (*Стрелка влево = #75.*)
      if ch=#77 then getkey='RIGHT';        (*Стрелка вправо = #77.*)
      if ch=#82 then getkey= INSERT',       (*Клавиша insert = #82.*)
      if ch=#83 then getkey='DELETE ;       (*Клавиша delete = #83.*)
      if ch=#71 then getkey='HOME';         (*Клавиша home = #71.*)
      if ch=#79 then getkey='END';          (*Клавиша end = #79.*)
  end
      else
          (*Если нажата не дополнительная клавиша.*)
          begin
              if ch=#13 then getkey.='RETURN'
              else if (ch=8) or (ch=127)    then getkey.='BACKSPACE'
                                            else getkey=ch;
          end;
end;
```

```
Procedure init_8253;
begin
(*Управляющее слово = b6h = 10110110b:
  10 = выбор счетчика 2;
  11 = сначала чтение/запись младшего байта, затем старшего;
  011 = режим 3;
  0 = шестнадцатибитовый двоичный счет.*)
  port($43):=$b6;             (*Запись слова в управляющий регистр 8253.*)
  port($42):=255;             (*Загрузка младшего байта.*)
  port($43):=255;             (*Загрузка старшего байта.*)
  port($61):=port($61) or 1   (*Выключение внутреннего динамика.*)
  port($43):=$80              (*80h - команда фиксации для счетчика 3.*)
end;

Function read_8253:integer;
(*Считывание младшего и старшего байтов счетчика.*)
var
  low_byte,high_byte:byte;
begin
  low_byte:=port($42);
  high_byte:=port($43);
  read_8253:=low_byte+256*high_byte;
end;

function find_period(address:integer; bit_weight:integer):real;
(* Нахождение длительности периода входного цифрового сигнала.
  Входной сигнал определяется адресом порта ввода и номером бита:
  Бит 0 - bit_weight=1
  Бит 1 - bit_weight=2
  ...
  Бит 7 - bit_weight=128.*)
var
  count,average_number,time1,time2:integer;
begin
  (*Определение промежутка времени, в течение которого входной сигнал имеет низкий уровень.
  Это значение будет использоваться для вычисления переменной average_number.*)
  repeat until port(address) and bit_weight=bit_weight;     (*Высокий уровень.*)
  repeat until port(address) and bit_weight=0;              (*Низкий уровень.*)
  time1:=read_8253;        (*Считывание первого значения счетчика.*)
  repeat until port(address) and bit_weight=bit_weight;     (*Снова высокий уровень.*)
  time2:=read_8253;        (*Считывание второго значения счетчика.*)
  average_number:=round(10000/(time2-time1));               (*Определение average_number.*)
  if average_number=0 then average_number:=1;
  repeat until port(address) and bit_weight=bit_weight;     (*Высокий уровень.*)
  repeat until port(address) and bit_weight=0;              (*Низкий уровень.*)
  time1:=read_8253;        (*Считывание первого значения счетчика.*)
  for count:=1 to average_number do   (*Нахождение спада входного сигнала.*)
     begin
       repeat until port(address) and bit_weight=bit_weight;      (*Высокий уровень.*)
       repeat until port(address) and bit_weight=0;               (*Низкий уровень.*)
     end;
```

```
  time2:=read_8253;       (*Считывание второго значения счетчика.*)
  find_period:=((time1-time2)*1/2*1193180)*1e6/average_number);
end;
```

Библиотека ресурсов WLIB.PAS

```
(*Программная библиотека ресурсов A.*)
(*Процедуры ввода/вывода для параллельного, последовательного и игрового портов.*)

Library LibA;

uses
Wincrt;

(*-Библиотека ресурсов № A1 (определение базового адреса LPT) -*)
Function Centronic(x:integer):integer; export;
(* $000:$0408 содержит базовый адрес для LPT1,
  $000:$040A содержит базовый адрес для LPT2,
  $000:$040C содержит базовый адрес для LPT3,
  $000:$040B содержит базовый адрес для LPT4,
  $000:$0411 содержит количество параллельных портов.*)
var
  number_of_LPT,LPT1,LPT2,LPT3,LPT4:integer;
begin
     number_of_lpt:=mem($0000:$0411);     (*Считывание количества установленных параллельных
                                           портов *)
     number_of_lpt:=(number_of_lpt and (128+64)) shr 6;
     lpt1:=0; lpt2:=0; lpt3:=0; lpt4:=0;
     lpt1:=memw($0000:$0408);              (*Процедура считывания из памяти.*)
     lpt2:=memw($0000:$040A);
     lpt3:=memw($0000:$040C);
     lpt4:=memw($0000:$040E);
     case x of
         0: centronic:=number_of_LPT;
         1: centronic:=LPT1;
         2: centronic:=LPT2;
         3: centronic:=LPT3;
         4: centronic:=LPT4;
     end;
end;
(*-Библиотека ресурсов № A2 (нахождение веса бита).-*)
Function bit_weight(bit:integer):integer; export;
var
  i,dummy:integer;
begin
  if bit=1 then bit_weight:=1
  else
     begin
         dummy:=1
         for i:=1 to bit-1 do dummy:=dummy*2;
         bit_weight:=dummy;
     end;
end;
```

```
(*-Библиотека ресурсов № A3 (считывание данных в компьютер) -*)
Function Read_status_port(P_address integer) integer, export,
var
  byte1 byte,
begin
  byte1 =port(P_address+1),        (*Считывание байта из регистра состояния *)
  byte1 =byte1 and 120,            (*01111000 (от старшего к младшему) and 0dddd  =
                                   0dddd000 *)
  Read_status_port =byte1 shr 3,   (*Сдвиг на 3 бита вправо, Read_status_port=0000hhhh *)
end,

(*- Библиотека ресурсов № A4 (запись в регистр данных) -*)
Function Write_data_port(P_address integer,port_data integer) integer, export,
(*Биты регистра данных не инвертированы *)
begin
  port(P_address) =port_data,      (*Ввод байта в регистр данных *)
end

(*-Библиотека ресурсов № A5 (запись данных в регистр управления компьютера) -*)
Function Write_control_port(P_address integer, port_data integer) integer, export,
(*Биты 0, 1 и 3 инвертированы  Требуются преобразования над битами *)
begin
  if port_data and 1=1 then port_data =port_data and (255-1)
     else port_data =port_data or 1,
  if port_data and 2=2 then port_data =port_data and (255-2)
     else port_data =port_data or 2
  if port_data and 8=8 then port_data =port_data and (255-8)
     else port_data =port_data or 8
  port(P_address+2) =port_data,    (*Ввод байта данных в порт управления *)
end

(*-Библиотека ресурсов № A6 (определение базовых адресов COM-портов) -*)
Function RS232(x integer) integer, export,
{Универсальное средство определения адресов COM-портов }
(* $0000 $0400 содержит базовый адрес порта COM1,
  $0000 $0402 содержит базовый адрес порта COM2,
  $0000 $0404 содержит базовый адрес порта COM3,
  $0000 $0406 содержит базовый адрес порта COM4,
  $0000 $0411 содержит количество COM-портов *)
var
number_of_COM COM1 COM2,COM3,COM4 integer,
begin
  Number_of_COM =mem($0000 $0411)      (*Считывание количества COM-портов *)
  Number_of_COM =(Number_of_COM and (8+4+2)) shr 1,
  COM1 =0, COM2 =0, COM3 =0, COM4 =0,
  COM1 =memw($0000 $0400),             (*Процедура считывания из памяти *)
  COM2 =memw($0000 $0402),
  COM3 =memw($0000 $0404),
  COM4 =memw($0000 $0406),
```

```
  Case x of
      0: RS232:=number_of_COM;
      1: RS232:=COM1;
      2: RS232:=COM2;
      3: RS232:=COM3;
      4: RS232:=COM4;
end;

(*-Библиотека ресурсов № A7 (запись в регистр разрешения прерываний).-*)
Function write_interrupt_enable(RS232_address, output_byte:integer); export;
begin
  port(RS232_address+1):=Output_byte;
end;

(*-Библиотека ресурсов № A8 (чтение из регистра идентификации прерываний).-*)
Function read_interrupt_identification(RS232_address:integer):integer; export;
begin
  read_interrupt_identification:=port(RS2132_address+2);
end;

(*-Библиотека ресурсов № A9 (запись в регистр формата данных).-*)
Function Write_data_format(RS232_address, Baudx, Parity, Data_bit, Stop_bit:integer):integer;
export;
var
  byte1,byte2,output_byte:byte;
  divisor:integer;
  baud:longint;
begin
  baud:=baudx*100;
  divisor:=115200 div Baud;
  if divisor<=255 then begin byte1:=divisor; byte2:=0; end;
  if divisor>255 then begin byte1:=divisor mod 256; byte2:=divisor div 256; end;
  output_byte:=(data_bit-5)+4*(stop_bit-1)+8*parity;
  port(RS232_address+3):=128;           {Загрузка формата последовательных данных,
                                        первый бит регистра равен 1.}
  port(RS232_address+0):=byte1;         {Младший байт делителя равен 1.}
  port(RS232_address+1):=byte2;         {Старший байт делителя равен 0.}
  port(RS232_address+3):=output_byte;   {Загрузка делителя и других параметров.}
end;

(*-Библиотека ресурсов № A10 (запись данных в буферный регистр передатчика).-*)
Function write_transmit_buffer(RS232_address,output_byte:integer):integer; Export
begin
  port(RS232_address):=output_byte;
end;

(*-Библиотека ресурсов № A11 (запись данных в регистр состояния модема).-*)
Function write_modem_status(RS232_address, RTS, DTR:integer):integer;export;
(*RTS и DTR инвертируются с помощью MAX238 на экспериментальной плате.*)
(*RTS=бит 1, DTR=бит 0 регистра управления модемом, смещение 04h.*)
begin
  RTS:=1-RTS;
```

```
  DTR:=1-DTR;
  port(RS232_address+4):=RTS*2+DTR;        (*Запись в регистр 04h.*)
end;

(*-Библиотека ресурсов № A12 (чтение данных из буферного регистра приемника).-*)
Function read_receive_buffer(RS232_address:integer):integer;export;
begin
  read_receive_buffer:=port(RS232_address);
end;

(*-Библиотека ресурсов № A13 (чтение данных из регистра состояния модема).-*)
Function read_modem_status(RS232_address, x:integer):integer; export;
(*x=1 - выбор бита DCD, x=2 - выбор бита DSR, x=3 - выбор бита CTS.*)
(*DCD=бит 7, DSR=бит 5, CTS=бит 4 регистра состояния модема, смещение 06h.*)
(*Все биты инвертируются с помощью MAX238 на экспериментальной плате.*)
var
  input_byte:byte;
begin
  input_byte:=port(RS232_address+6);
  case x of
     1:  Read_modem_status:=1-round((input_byte and 128)/128);
     2:  Read_modem_status:=1-round((input_byte and 32)/32);
     3:  Read_modem_status:=1-round((input_byte and 16)/16);
  end;
end;

(*-Библиотека ресурсов № A14 (чтение регистра игрового порта).-*)
Function read_game_port(bitx:integer):integer;export
(*Адрес игрового порта: 201h.
Bitx (1-8) выбирает состояние AX. AY, BX. BY, BA1, BA2, BB1 и BB2.*)
var
  input_byte:byte;
begin
  input_byte:=port($201);
  read_game_port:=round((input_byte and bit_weight(bitx))/bit_weight(bitx));
end;

(*-Библиотека ресурсов № A15 (запись в регистр игрового порта).-*)
Function write_game_port:integer; export;
(*Вводит 0 в игровой порт для запуска одновибратора.*)
begin
  port($201):=0;
end;

(*-Библиотека ресурсов № A16 (получение длительности периода одновибратора).-*)
Function interval_game_port(x:integer):integer; export;
(*x выбирает AX(x=1), AY(x=2), BX(x=3), BY(x=4).*)
var
  time1,time2,dummy:integer;

Procedure init_8253;
(*Настройка 8253.*)
begin
```

```
(*Управляющее слово = b6h = 10110110b:
   10 = выбор счетчика 2;
   11 = сначала чтение/запись младшего байта, затем старшего;
   011 = режим 3;
   0 = шестнадцатибитовый двоичный счет.*)
   port($43):=$b6;               (*Запись в управляющий регистр 8253.*)
   port($42):=255;               (*Загрузка младшего байта.*)
   port($43):=255;               (*Загрузка старшего байта.*)
   port($61):=port($61) or 1;    (*Выключение внутреннего динамика.*)
   port($43):=$80;               (*80h - команда фиксации для счетчика 3.*)
end;

Function read_8253:integer;
(*Считывание младшего и старшего байтов счетчика.*)
var
   low_byte,high_byte:byte;
begin
   low_byte:=port($42);
   high_byte:=port($43);
   read_8253:=low_byte+256*high_byte;
end;

var
   i:integer;
begin
   init_8253;
   for i:=1 to 10 do i:=i;
   i:=0;
   dummy:=bit_weight(x);
   port($201):=0;
   time1:=read_8253;
   repeat i:=i+1 until (port($201) and dummy=0) or (i>=5000);
   time2:=read_8253;
   interval_game_port:=time2-time1;
   if i>=10000 then interval_game_port:=0;
end;

exports
   centronic                        index 1,
   bit_weight                       index 2,
   read_status_port                 index 3,
   write_data_port                  index 4,
   write_control_port               index 5,
   RS232                            index 6,
   Write_interrupt_enable           index 7,
   Read_interrupt_identification    index 8,
   Write_data_format                index 9,
   write_transmit_buffer            index 10,
   write_modem_status               index 11,
   read_receive_buffer              index 12,
   read_modem_status                index 13,
   read_game_port                   index 14,
   write_game_port                  index 15,
   interval_game_port               index 16;
begin
end.
```

4. РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ПАРАЛЛЕЛЬНОГО, ПОСЛЕДОВАТЕЛЬНОГО И ИГРОВОГО ПОРТОВ

Для многих приложений недостаточно того количества линий ввода/вывода, которое обеспечивают стандартные порты, и требуется увеличение их числа.

4.1. Расширение возможностей параллельного порта

Одним из самых простых и экономичных способов увеличения количества линий ввода/вывода является использование цифровых микросхем ТТЛ и КМОП серий 74LS и 4000 с малой степенью интеграции. Если же вы хотите изменять конфигурацию устройств, потребуются специализированные программируемые микросхемы повышенной степени интеграции – 8255, 8155 и др. Например, микросхема 8255 позволяет формировать 24-канальный ввод/вывод, который разделен на три группы: А, В и С. Каждая группа имеет восемь линий ввода/вывода и может быть настроена как на ввод, так и на вывод.

4.1.1. Увеличение количества линий ввода/вывода при помощи микросхем с малой степенью интеграции

В разделе приводятся принципиальные схемы для ввода/вывода данных с использованием цифровых микросхем с малой степенью интеграции, а также программы управления к ним.

Чтение восьмибитовых данных

На рис. 4.1 изображены микросхема 74LS241 и принципиальная схема, позволяющая считывать восьмибитовые данные через параллельный порт компьютера. Когда на контакт 1 подается низкий уровень, открываются четыре левых буфера, а когда на контакт 19 поступает высокий уровень – четыре правых. Если контакты 1 и 19 соединены, то, последовательно подавая на них высокий и низкий уровни, можно считать четыре бита в буферы слева, затем четыре бита в буферы справа и наоборот, то есть восемь бит считываются при помощи только четырех линий.

Соединенные таким образом контакты 1 и 19 образуют линию выборки данных (DSL), которая управляется одним из битов регистра данных.

На рис. 4.1 также показано соединение микросхемы с экспериментальной платой параллельного порта. Контакты D1, S1 – S4 находятся на плате. Восьмибитовые входные данные считываются в компьютер за два последовательных обращения к порту. Когда на линии DSL низкий уровень, считываются биты 0–3, а когда высокий – биты 4–7. Чтобы получить исходный байт (биты 0–7), необходимо

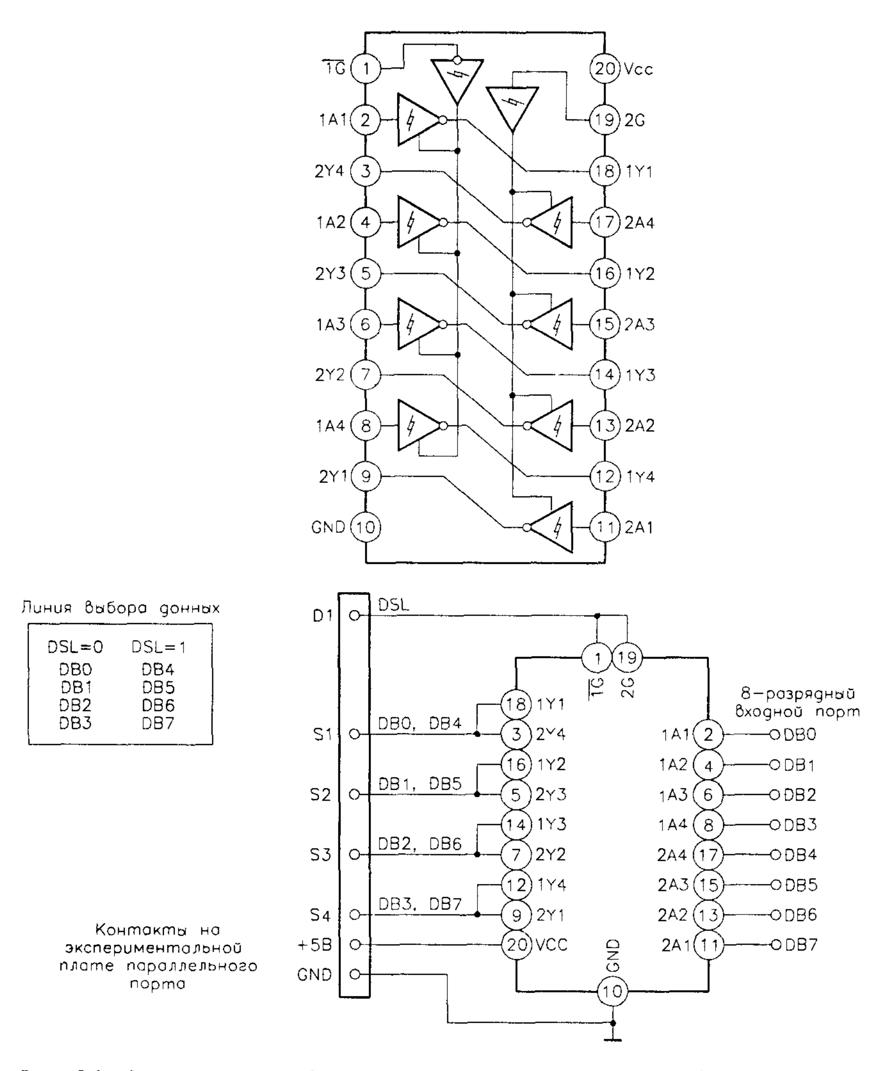
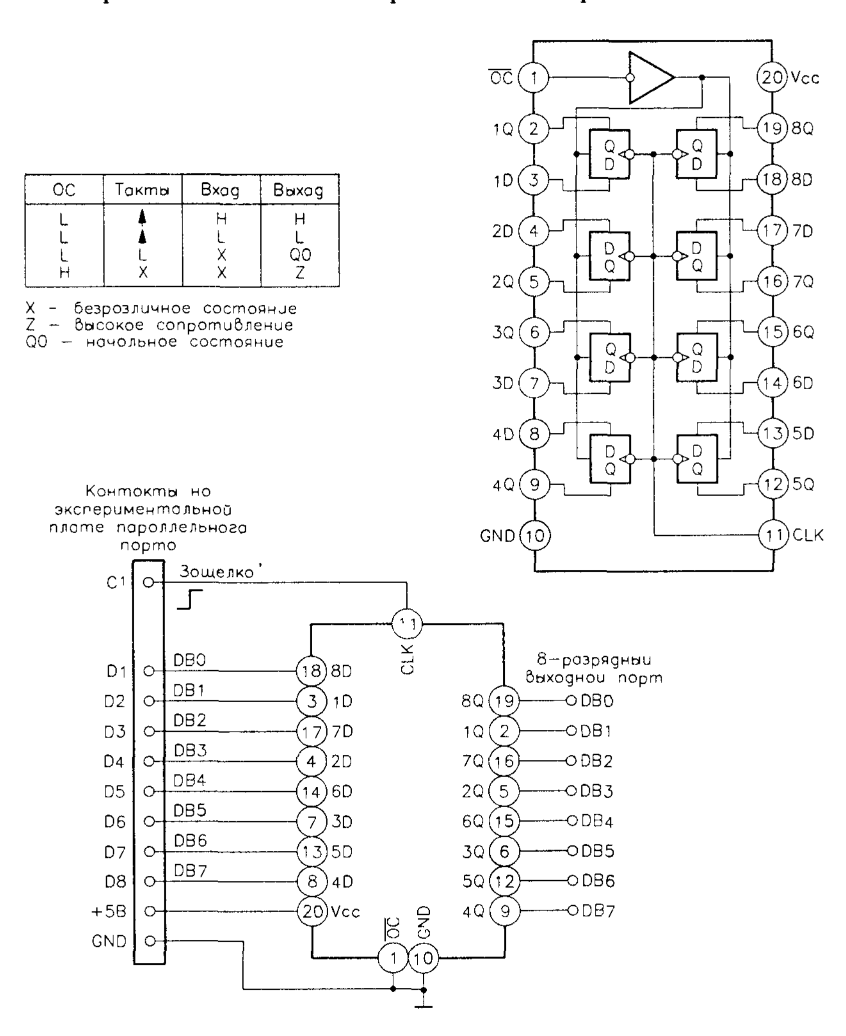


Рис. 4.1. Схема для чтения байта данных через четырехбитовый цифровой вход при помощи микросхемы 74LS241

произвести несложные манипуляции над битами. Проверить работу схемы можно, используя программу управления экспериментальной платой (см. главу 3).

Вывод данных

Для вывода данных применяются *регистры-защелки*, например 74LS373 или 74LS374. Входы 74LS374 (рис. 4.2) соединены с битами 0–7 регистра данных. Запись данных в микросхеме управляется контактом CLK (контакт 11) и осуществляется по положительному фронту. Логическое состояние на входе CLK зависит от состояния выходной линии регистра управления. Контакты C1, D1 – D8 находятся на экспериментальной плате параллельного порта.



OC	Такты	Вход	Выход
L	↑	H	H
L	↑	L	L
L	L	X	Q0
H	X	X	Z

Рис. 4.2. Схема для вывода данных с использованием микросхемы 74LS374

Текст программы LS374.PAS

```
Program Centronic_output_expander_using_74LS374;
(*Программа для расширения возможностей порта при помощи 74LS374 *)
uses
  graph,crt,dos;
var
  ch:char;
(*Подключение двух библиотек: TPLIB1 и TPLIB2.*)
{$I c:\ioexp\tplib1.pas}
{$I c:\ioexp\tplib2.pas}
procedure Load_data_to_LS374,
(*Загрузка данных в 74LS374.*)
var
  output_byte.byte;
begin
  (*Ввод данных в порт данных.*)
  write_control_port(P_address,0);
  write('Input the output data: '); readln(output_byte);
  write_data_port(P_address,output_byte);
  write('Press RETURN to load the input data to the output of 74LS374');
  readln:
  write_control_port(P_address,1);
  writeln('Line 1 of the Control port goes from low to high to latch data');
  delay(2000);
  write_control_port(P_address,0);
end;
(*Главная программа.*)
begin
  centronic_address,  (*Получение адреса параллельного порта.*)
  repeat
    clrscr,
    load_data_to_LS374;
    write('Continue (Y/N): '); readln(ch);
  until upcase(ch)='N';
end.
```

4.1.2. Увеличение количества линий ввода/вывода при помощи микросхемы 8255

На рис. 4.3 приведены выводы и внутренняя блок-схема микросхемы 8255 с 24 линиями ввода/вывода, разделенными на три восьмиразрядных порта: A, B и C.

Микросхема также имеет четыре внутренних регистра. Три из них называются периферийными и относятся к портам A, B и C. Четвертый – это регистр управления. Периферийные регистры используются для передачи данных через порт, а регистр управления – для хранения параметров настройки. Он представляет собой восьмиразрядную двунаправленную шину (биты 0–7, контакты 34–27), через которую данные считываются или записываются под управлением линий $\overline{WR}$ (контакт 36) и $\overline{RD}$ (контакт 5). Линии адреса A0 (контакт 9) и A1 (контакт 8) служат для выбора внутреннего регистра:

- A0=0, A1=0 – выбор регистра A;
- A0=1, A1=0 – выбор регистра B;

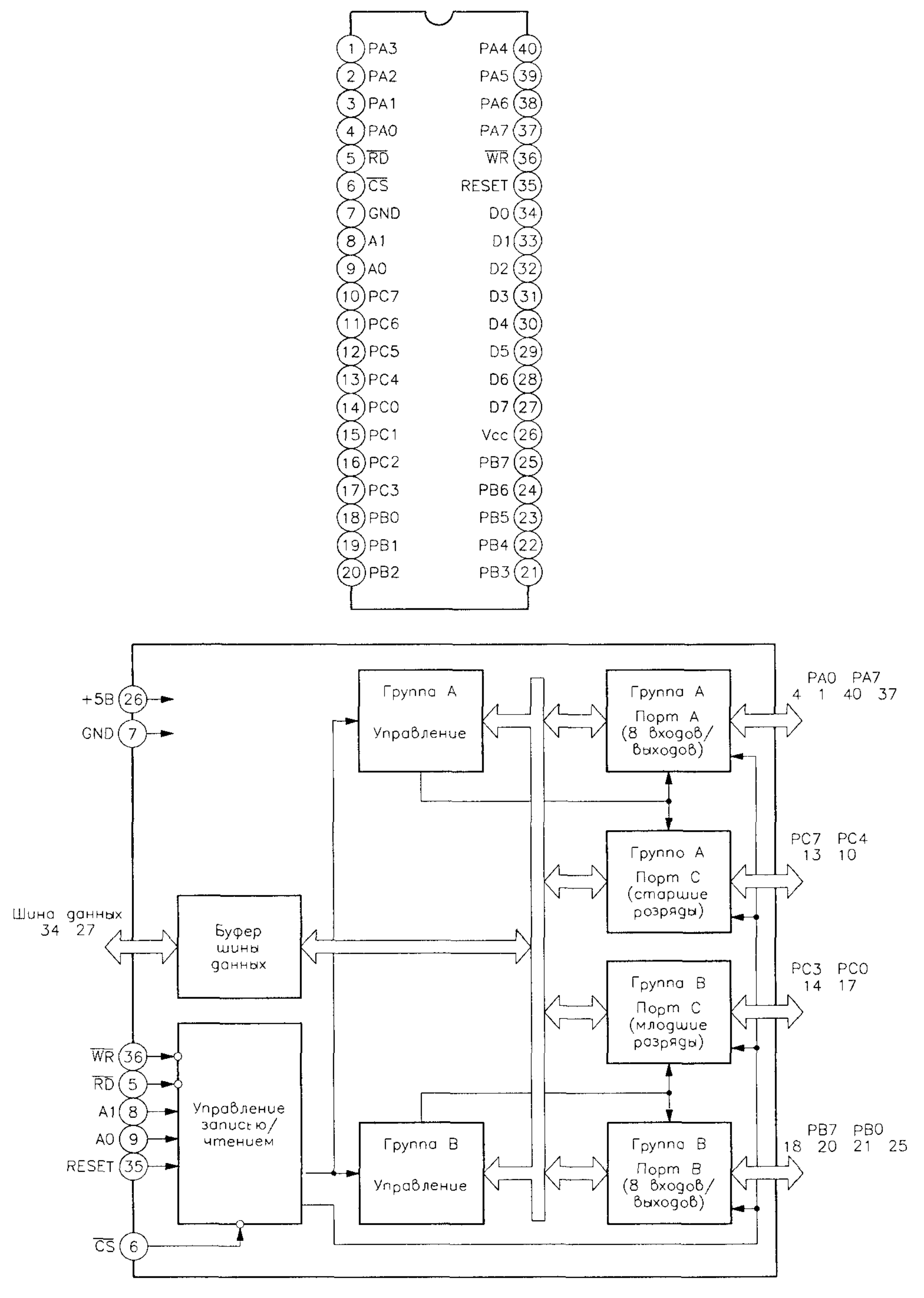


Рис. 4.3. Выводы и внутренняя блок-схема микросхемы 8255

- A0=0, A1=1 – выбор регистра C;
- A0=1, A1=1 – выбор регистра управления.

Для нормального функционирования микросхемы на линию $\overline{CS}$ (контакт 6) должен подаваться низкий уровень. Линия RESET (контакт 35) предназначена для сброса микросхемы, который осуществляется высоким уровнем. После сброса все линии ввода/вывода портов A, B и C настраиваются как входы. В нормальном режиме линия RESET должна удерживаться в состоянии логического нуля.

Микросхема имеет три режима работы:

- режим 1: порты A и B настраиваются как восьмиразрядные порты ввода/вывода. Смешивание линий ввода и вывода не допускается. Порт C делится на две половины (четыре старших и четыре младших бита). Каждая половина может быть настроена как на вход, так и на выход. Смешивание линий ввода и вывода в пределах одной половины запрещено;
- режим 2: микросхема настраивается как управляемый порт ввода/вывода, порты A и B – как независимые порты ввода или вывода. Каждый из них имеет 4-разрядный порт управления, который формируется в порте C соответственно как четыре младших и старших бита;
- режим 3: порт A настраивается как двунаправленный порт.

Режимы работы микросхемы 8255 устанавливаются с помощью регистра управления, куда необходимо записать управляющее слово. Назначение битов управляющего слова приведено ниже:

бит 7 (флаг установки режима)	всегда единица
биты 6, 5 (биты выбора режима)	00 = режим 1
	01 = режим 2
	1x = режим 3
бит 4 (режим порта A)	1 = вход
	0 = выход
бит 3 (режим верхней половины порта C)	1 = вход
	0 = выход
бит 2 (выбор подрежима для режима 3)	1 = режим 1
	0 = режим 0
бит 1 (режим порта B)	1 = вход
	0 = выход
бит 0 (режим нижней половины порта C)	1 = вход
	0 = выход

Схема соединения микросхемы 8255 с параллельным портом показана на рис. 4.4.

Регистр данных LPT-порта передает данные в микросхему 8255, этим процессом управляют DD2 и DD3 (буферы с тремя состояниями 74LS241 и 74LS244). Регистр состояния считывает данные из 8255. Две линии порта управления соединены с линиями адреса 8255, а две другие – с $\overline{RD}$ и $\overline{WR}$.

Требуемые данные сначала записываются в регистр данных, а состояние линий A0 и A1 – в регистр управления. Затем по линии $\overline{WR}$ подается нулевой

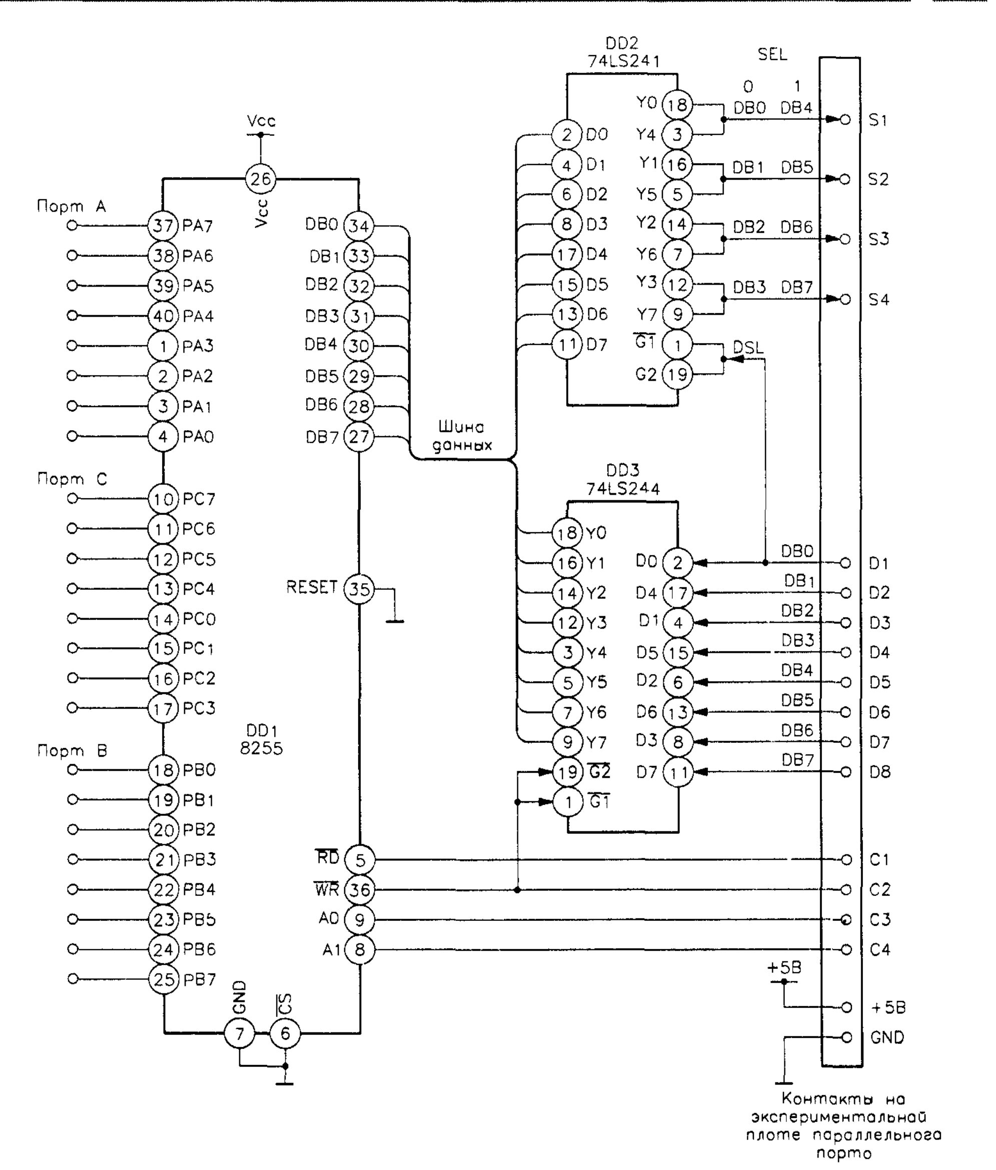


Рис. 4.4. Схема с использованием программируемого устройства ввода/вывода параллельной информации 8255

импульс (перепад 1–0–1), при котором открываются буферы данных DD3. Перепад 1–0 записывает информацию в выбранный регистр 8255. Данные из микросхемы 8255 считываются DD2 (74LS241). Линия выбора данных (DSL) управляется битом 0 порта данных. При считывании данных состояние линий A0 и A1 сначала записывается в регистр управления, а линия $\overline{RD}$ удерживается

в нулевом состоянии. При этом 8255 выводит данные на собственную шину данных. Затем линия DSL первый раз переходит в ноль, и регистр состояния производит первое считывание. После этого линия DSL снова устанавливается в единицу, и производится второе считывание. Таким образом, за два считывания восстанавливается исходный байт данных.

Управляющая программа написана на языке Turbo Pascal 6. Она настраивает все порты как выходы (режим 0, управляющее слово 128) и содержит все необходимые процедуры и функции, позволяющие на их основе разрабатывать собственное программное обеспечение. `Write_control(Control_word:byte)` записывает управляющее слово во внутренний регистр управления 8255. `Write_port (port_number,output_byte:byte)` записывает байт в порт A, B или C, который был настроен как выход. `Inputbyte(port_number:byte):byte` считывает данные из порта A, B или C, настроенного как вход.

Текст программы 8255.PAS

```
Program Centronic_8255_interface;
(*Некоторые используемые управляющие слова:
   128(десятичное): порты A, B и C - выходы;
   155(десятичное): порты A, B и C - входы;
   144(десятичное): порт A - вход; порты B и C - выходы;
   146(десятичное): порты A и B - входы; порт C - выход.*)
uses
   graph,crt,dos;
var
   command,output_byte,input_byte,bitnumber,portnumber,outputbyte:byte;
   P_address,delaynumber:integer;
{$I c:\ioexp\tplib1.pas}
procedure find_delay_number;
(*Определение скорости компьютера и количества тактов за 1 мс.*)
var
   time1,time2,dt:real;
   t,h1,m1,s1,s1001,h2,m2,s2,s1002:word;
begin
   clrscr;
   gotoxy(25,24); write('Checking computer speed');
   gettime(h1,m1,s1,s1001);
   time1:=3600*h1+60*m1+s1+s1001/100;
   for t:=1 to 1000 do delay(1);
   gettime(h2,m2,s2,s1002);
   time2:=3600*h2+60*m2+s2+s1002/100;
   dt:=time2-time1;
   delaynumber:=round(1000/dt*0.001);
   clrscr;
   gotoxy(30,24); write('Finished...');
   clrscr;
end;

Function convert(bytex:byte):byte;
(*Преобразование bytex в байт для передачи в регистры 8255.*)
(* Байт, принимаемый 8255: b8 b7 b6 b5 b4 b3 b2 b1,
```

```
  баит  выходящии с параллельного порта  b8 b4 b7 b3 b6 b2 b5 b1
  Разница в битовых последовательностях получается из за соединения с платои  Для передачи
  данных в 8255 необходимо преобразовать их в соответствующую форму *)
begin
  convert =    ((bytex and 1)+((bytex and 2) shl 1)+
               ((bytex and 4) shl 2)+((bytex and 8) shl 3)+
               ((bytex and 16) shr 3)+((bytex and 32) shr 2)+
               ((bytex and 64) shr 1)+(bytex and 128)
end

procedure write_control(control_word byte)
(*Запись управляющего слова в регистр управления ИС 8255 *)
(*Управляющее слово вводится из порта [P_address]  чтение/запись 8255 управляется портом
[P_address+2] *)
(*Конфигурация битов порта [P_address+2] -1(Read) -2(Write) 3(A0)  4(A1) *)
begin
  control_word =convert(control_word)                 (*Преобразование control_word *)
  port(P_address) =control_word  delay(delaynumber*10)  (*Ввод управляющего слова из порта
                                                      [P_address] *)
  port(P address+2) -0+0+4+0  delay(delaynumber*3)     (*Read=1  Write=1  A0=1  A1=1
                                                      задержка=10 мс *)
  port(P_address+2) =0+2+4+0  delay(delaynumber*3)     (*Read=1  Write=0  A0=1  A1=1
                                                      задержка=10 мс *)
  port(P_address+2) =0+0+4+0  delay(delaynumber*3)     (*Read=1  Write-1  A0=1  A1=1
                                                      задержка=10 мс *)
end

procedure write_port(port_number output_byte byte)
(*Запись баита (output_byte) в порт ИС 8255  определенныи переменнои port_number *)
(*Выходной байт вводится из порта [P_address]  чтение/запись 8255 управляется портом
[P address+2] *)
(*Конфигурация битов порта [P_address+2] -1()  2() 3(A0)  4(A1) *)
begin
  output_byte =convert(output_byte)                   (*Преобразование output_byte *)
  port(P_address) =output_byte                        (*Ввод баита из порта [P_address] *)
  if port_number=0 then    begin                      (*Ввод баита в регистр A ИС 8255 *)
                               port(P_address+2) =0+0+0+8  delay(delaynumber*3)
                                                      (*Read=1  Write=1  A0=1  A1=0 *)
                               port(P_address+2) =0+2+0+8  delay(delaynumber*3)
                                                      (*Read=1  Write=0  A0=1  A1-0 *)
                               port(P_address+2) =0+0+0+8  delay(delaynumber*3)
                                                      (*Read=1  Write=1  A0=1  A1=0 *)
                           end
if port_number=1 then      begin                      (*Ввод баита в регистр B ИС 8255 *)
                               port(P_address+2) =0+0+4+8  delay(delaynumber*3)
                               port(P_address+2) =0+2+4+8  delay(delaynumber*3)
                               port(P_address+2) =0+0+4+8  delay(delaynumber*3)
                           end
if port_number=2 then      begin                      (*Ввод баита в регистр C ИС 8255 *)
                               port(P_address+2) =0+0+0+0  delay(delaynumber*3)
                               port(P_address+2) =0+2+0+0  delay(delaynumber*3)
                               port(P_address+2) =0+0+0+0  delay(delaynumber*3)
                           end
end
```

```
function inputbyte(port_number:byte):byte;
(*Считывание байта (input_byte) из порта микросхемы 8255, определенного переменной port_number.*)
(*Байт вводится из порта [P_address+1], восьмибитовые данные вводятся в параллельный порт
в два этапа. На первом этапе считываются биты 1, 2, 3 и 4; на втором - биты 5, 6, 7 и 8.
Эта операция управляется битом 1 порта [P_address]. Чтение/запись 8255 управляется портом
[P_address+2].*)
(*Конфигурация битов порта [P_address+2]:-1(Read) -2(Write) 3(A0) -4(A1).*)
var
  byte_1st,byte_2nd:byte;
begin
  if port_number=0 then   begin                         (*Чтение данных из порта A 8255.*)
                            port(P_address):=0;         (*DSL=0, подготовка к считыванию
                                                        младших четырех бит.*)
                            port(P_address+2):=0+0+0+8; delay(delaynumber*3);
                                                        (*Read=1, Write=1, A0=0, A1=0.*)
                            port(P_address+2):=1+0+0+8; (*Read=0, Write=1, A0=0, A1=0.*)
                            byte_2nd:=port(P_address+1); (*Считывание младших
                                                        четырех бит.*)
                            port(P_address):=1;         (*DSL=1, подготовка к считыванию
                                                        старших четырех бит.*)
                            byte_1st:=port(P_address+1); (*Считывание старших
                                                        четырех бит.*)
                            port(P_address+2):=0+0+0+8; delay(delaynumber*3);
                                                        (*Read=1, Write=1, A0=0, A1=0.*)
                          end;
  if port_number=1 then   begin                         (*Чтение данных из порта B 8255.*)
                            port(P_address):=0;
                            port(P_address+2):=0+0+4+8; delay(delaynumber*3);
                            port(P_address+2):=1+0+4+8;
                            byte_2nd:=port(P_address+1);
                            port(P_address):=1,
                            byte_1st:=port(P_address+1);
                            port(P_address+2).=0+0+4+8; delay(delaynumber*3);
                          end;
  if port_number=2 then   begin                         (*Чтение данных из порта C 8255.*)
                          port(P_address):=0;
                          port(P_address+2):=0+0+0+0, delay(delaynumber*3);
                          port(P_address+2).=1+0+0+0;
                          byte_2nd:=port(P_address+1);
                          port(P_address):=1;
                          byte_1st:=port(P_address+1);
                          port(P_address+2):=0+0+0+0; delay(delaynumber*3);
                        end;
  (*Примечание:   byte_1st=(x b8 b7 b6 b5 x x x)
                  byte_2nd=(x b4 b3 b2 b1 x x x)
                  x = любое число, b8 - b1 принимают значения 0 или 1.*)
  byte_1st:=byte_1st and 120;  (*x b8 b7 b6 b5 x x x and 01111000=0 b8 b7 b6 b5 0 0 0.*)
  byte_1st:=byte_1st shl 1;    (*Сдвиг на 1 бит влево, byte_1st=b8 b7 b6 b5 0 0 0 0.*)
  byte_2nd:=byte_2nd and 120;  (*x b4 b3 b2 b1 x x x and 01111000=0 b4 b3 b2 b1 0 0 0.*)
  byte_2nd:=byte_2nd shr 3;    (*Сдвиг на 3 бита вправо, byte_2nd=0 0 0 0 b4 b3 b2 b1.*)
end;
(*Главная программа.*)
begin
  find_delay_number;
```

```
clrscr,
Centronic_address,
repeat
  clrscr,
  writeln( Centronic port - 8255 card testing program ),
  writeln,
  writeln('Port A, B and C are all configured as Outputs ),
  writeln( The control word sent to the 8255 PPI is 128 decimal ),
  write_control(128), (*Порты A, B и C настроены как выходы *)
  write_control(128),
  writeln,
  write( Input the bit number to be tested (1 to 8, 9 to quit) ),
  readln(bitnumber),
  outputbyte =bit_weight(bitnumber),
  writeln,
  textcolor(yellow+blink),
  write( The selected bit of port A, B and C will change from 0 to 1 repeatedly ),
  write( You need a logic probe to test the card' ),
  textcolor(white),
  repeat
     write_port(0,outputbyte),
     write_port(1,outputbyte),
     write_port(2,outputbyte), delay(delaynumber*300),
     write_port(0,0),
     write_port(1,0)
     write_port(2 0), delay(delaynumber*300),
  until keypressed or (portnumber=9),
  readln,
until bitnumber=9,
end
```

4.2. Расширение возможностей последовательного порта

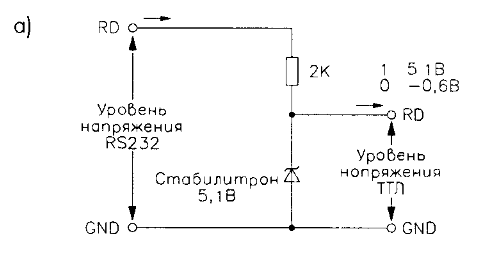
В разделе приводится информация о более сложных вариантах использования последовательного порта.

4.2.1. Преобразователи уровней RS232/ТТЛ

Наиболее просто преобразовать уровни напряжения RS232 в уровни ТТЛ с помощью полупроводниковых стабилитронов (рис. 4.5а). Схема состоит из одного резистора и пятивольтового стабилитрона. Когда на входе схемы высокий уровень напряжения, на выходе +5 В, в противном случае +0,6 В. Выходной сигнал может управлять схемами ТТЛ или КМОП.

Другая схема преобразователя RS232/ТТЛ представлена на рис. 4.5б. Она не требует внешнего источника питания (использует питание порта RS232) и инвертирует сигнал.

Достаточно широко применяются интегральные схемы преобразователей ТТЛ/ RS232, например MAX232 (Maxim RS655-290) и MAX238 (Maxim RS655-313) – см. рис. 4.6. Для работы обеих микросхем необходим источник питания



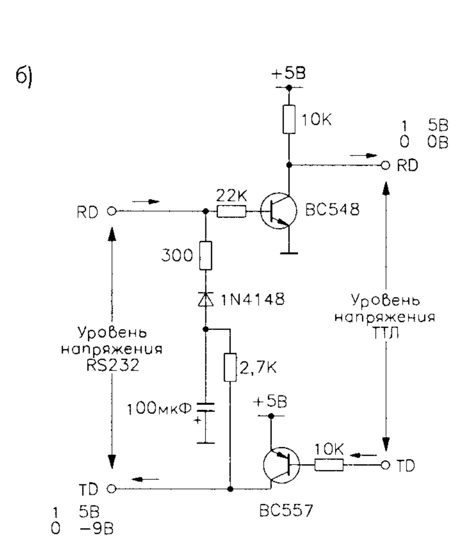


Рис. 4.5. Схемы преобразователей напряжения RS232/ТТЛ. а – RS232/ТТЛ на стабилитроне, б – RS232/ТТЛ и ТТЛ/RS232 на транзисторе

+5 В. Микросхема MAX232 имеет встроенные преобразователи напряжения, два преобразователя RS232/ТТЛ и два преобразователя ТТЛ/RS232. Преобразователь ТТЛ/RS232 превращает напряжение питания +5 В в напряжения +10 и –10 В. Следует помнить, что нельзя подключать выводы V+ и V– к потенциалу, меньшему 3 В по абсолютной величине (минимальный уровень для RS232). Когда на выходах V+ и V–, ток равен 20 мА, а напряжение составляет порядка +7 и –7 В. Входы преобразователей выдерживают напряжение до 25 В. Максимальная скорость передачи данных 120 Кб/с. Ток потребления микросхемы MAX232 в режиме холостого хода 4 мА. Характеристики MAX238 аналогичны MAX232, но она имеет по четыре преобразователя RS232/ТТЛ и ТТЛ/RS232.

Для автономных устройств с питанием от батарей может использоваться маломощная версия MAX3232 (Maxim RS189-1453), потребляющая только 250 мкА. Другие электрические характеристики аналогичны.

Чтобы обеспечить высокое шумоподавление и электрическую изоляцию между входом и выходом, применяются преобразователи с гальванической развязкой. Такие устройства, как NM232DD (Newport RS264-412) – см. рис. 4.7, имеют по два преобразователя RS232/ТТЛ и ТТЛ/RS232 и работают от напряжения питания +5 В. Напряжение, которое изоляция способна выдержать, равно 1500 В.

4.2.2. Увеличение количества линий ввода/вывода с помощью UART

UART CDP6402 (Harris Semiconductors, RS630-389) – это КМОП, *универсальный асинхронный приемопередатчик* для сопряжения с асинхронными каналами передачи последовательных данных (рис. 4.8). Микросхема имеет программируемый формат передачи данных, может передавать 5–8 бит данных и характеризуется наличием (или отсутствием) проверки на четность или нечетность, длительностью стоповой посылки 1, 1,5 или 2 бита. Напряжение питания составляет 4–10 В. Потребляемый ток 1,5 мА при напряжении питания +5 В.

Контакт 21 – вход для сброса микросхемы, на него обычно подается низкий уровень напряжения. Контакты 35–39 управляют форматом передачи данных.

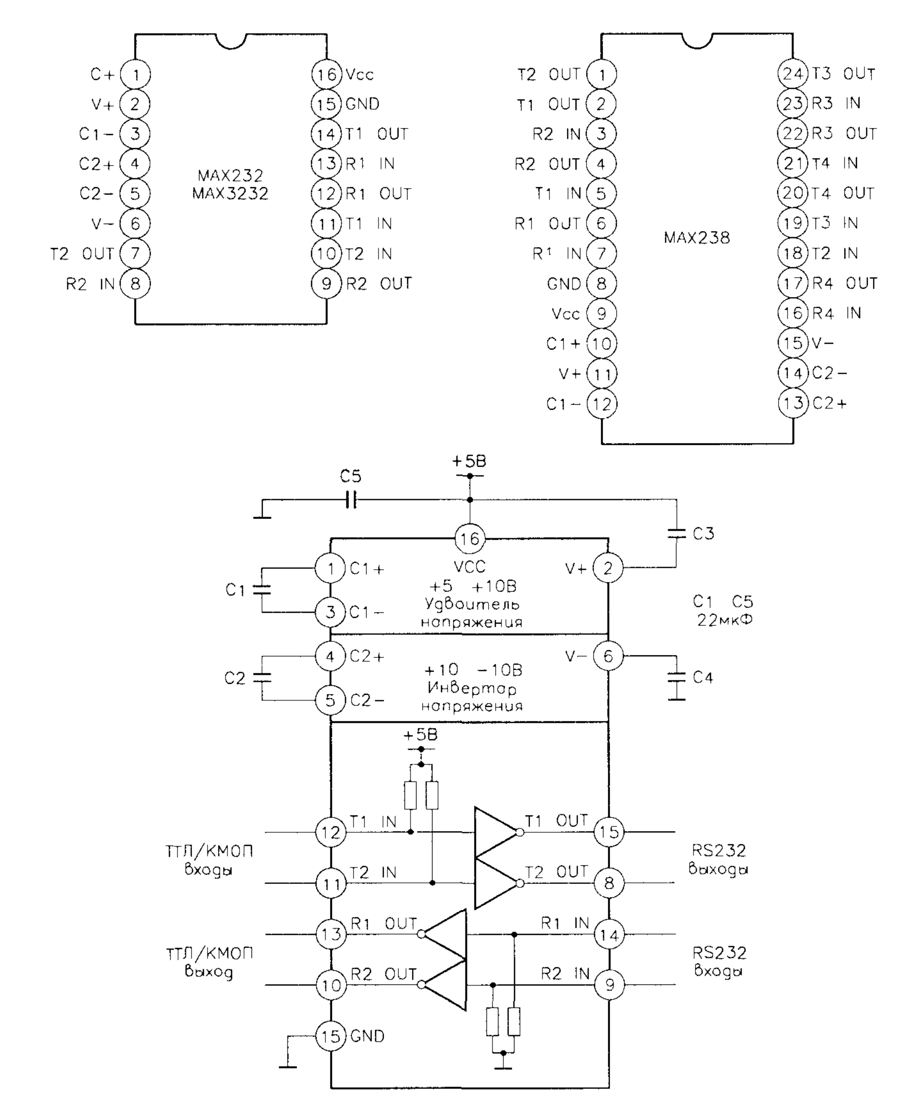


Рис. 4.6. Расположение выводов MAX232 и MAX238 и внутренняя блок-схема MAX232

Для работы с этими контактами необходимо подать высокий уровень напряжения на контакт 34 (управление регистром загрузки, CRL).

При подаче на контакт 35 (блокировка проверки, PI) высокого уровня блокируется проверка на четность. При этом на контакте 13 (ошибка паритета, PE) постоянно удерживается низкий уровень. Когда на входе PI ноль, для выбора проверки по четности необходимо на контакт 39 (разрешение проверки по четности, EPE) подать единицу. Если на EPE ноль, выбирается проверка по нечетности. Единица на контакте 36 (выбор длины стопового бита, SBS) дает длину стоповой посылки

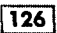


Рис. 4.7. Расположение и назначение выводов микросхемы NM232DD

1,5 бита для пятибитового формата и 2 бита для остальных форматов. Подача нуля дает один стоповый бит. Контакты 37 и 38 (выбор длины блока данных, соответственно CLS1 и CLS2) управляют установкой длины блока данных:

- CLS1 = 0, CLS2 = 0 – 5 бит;
- CLS1 = 1, CLS2 = 0 – 6 бит;
- CLS1 = 0, CLS2 = 1 – 7 бит;
- CLS1 = 1, CLS2 = 1 – 8 бит.

Контакты 17 (тактирование регистра приемника) и 40 (тактирование регистра передатчика) – это тактирующие входы приемника и передатчика, управляемые тактовыми импульсами, частота которых в 16 раз больше, чем требуемая скорость передачи. Обычно они соединены вместе.

Контакт 20 (вход регистра приемника) представляет собой вход для приема последовательных данных. Принятые данные преобразуются в параллельную

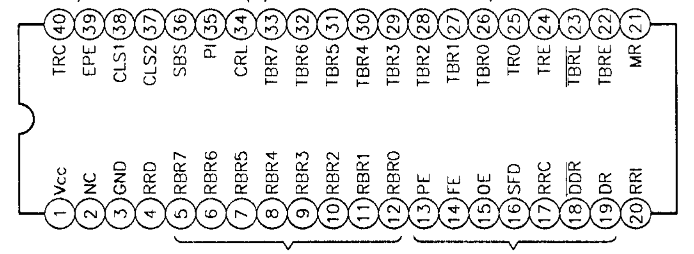


Рис. 4.8. Выводы и внутренняя блок-схема UART CDP6402

форму и сохраняются в буферном регистре приемника. Они доступны через контакты 5–12. На контакт 4 (запрещение приема, RRD) должен подаваться нуль. При условии нормального приема данных на контакт 19 (данные приняты) подается единица. Сюда допустимо подать нуль за счет подачи нуля на контакт 18 (сброс линии). После этого UART может принимать новые данные. Контакты 13 (ошибка паритета), 14 (ошибка блока) и 15 (ошибка переполнения) показывают наличие ошибок, возникших во время передачи. Чтобы получить доступ к этим выходам, необходимо на контакт 16 (запрещение доступа к флагам состояния, SFD) подать нуль.

Контакт 25 (выход регистра передатчика, TRO) – это выход последовательных данных. Параллельные данные сначала необходимо записать в буферные регистры передатчика (контакты 26–33). При подаче нуля на вывод 23 происходит загрузка данных в буферный регистр передатчика. Затем по положительному фронту они загружаются в регистр передатчика, и начинается собственно передача. Контакт 22 предназначен для индикации освобождения буферного регистра. Наличие здесь сигнала высокого уровня свидетельствует о том, что данные были переданы сюда, и буферный регистр готов к приему следующего блока данных. Контакт 24 служит для индикации освобождения регистра (регистр передатчика пуст). Наличие на нем логической единицы говорит о том, что передача данных завершена.

Временные диаграммы приема изображены на рис. 4.9а. Данные вводятся через вход RRI. Если данных нет, то на входе RRI должна быть единица. На этапе А на вход $\overline{DRR}$ подается нуль, при этом линия DR очищается. На этапе В во время первой стоповой посылки данные передаются из регистра приемника в буферный регистр приемника. Если линия DR не очищается перед началом передачи, то возникает ошибка переполнения. На этапе С, через 1/2 тактового интервала после этапа В, линия DR переходит в единичное состояние, показывая, что новые данные приняты. Единица на выходе FE означает, что принята неверная стоповая посылка; единица на выходе PE указывает на ошибку паритета. Если UART работает в непрерывном режиме, то $\overline{DRR}$ необходимо соединить с общим проводом.

Временные диаграммы передачи представлены на рис. 4.9б. На этапе А данные загружаются в буферный регистр передатчика через входы TBR0 – TBR7 по отрицательному фронту на входе $\overline{TBRL}$. Данные на входах TBR0 – TBR7 должны быть уже выставлены. Если длина блока данных менее восьми бит, то используются младшие биты. На этапе В положительный фронт по входу $\overline{TBRL}$ сбрасывает TBRE. После небольшой задержки данные передаются в регистр передатчика, а на выходе TRE появляется нуль. Выход TBRE переходит в единичное состояние, показывая, что буферные регистры передатчика пусты. Тактирующий сигнал для выходных данных должен иметь частоту, в 16 раз большую, чем скорость передачи. На этапе С $\overline{TBRL}$ переходит из единицы в нуль и затем снова в единицу, при этом в буферный регистр передатчика загружается второй блок данных. Передача данных в регистр передатчика задерживается до тех пор, пока не завершится передача текущего символа. На этапе D данные автоматически поступают в регистр передатчика и начинается передача второго блока.

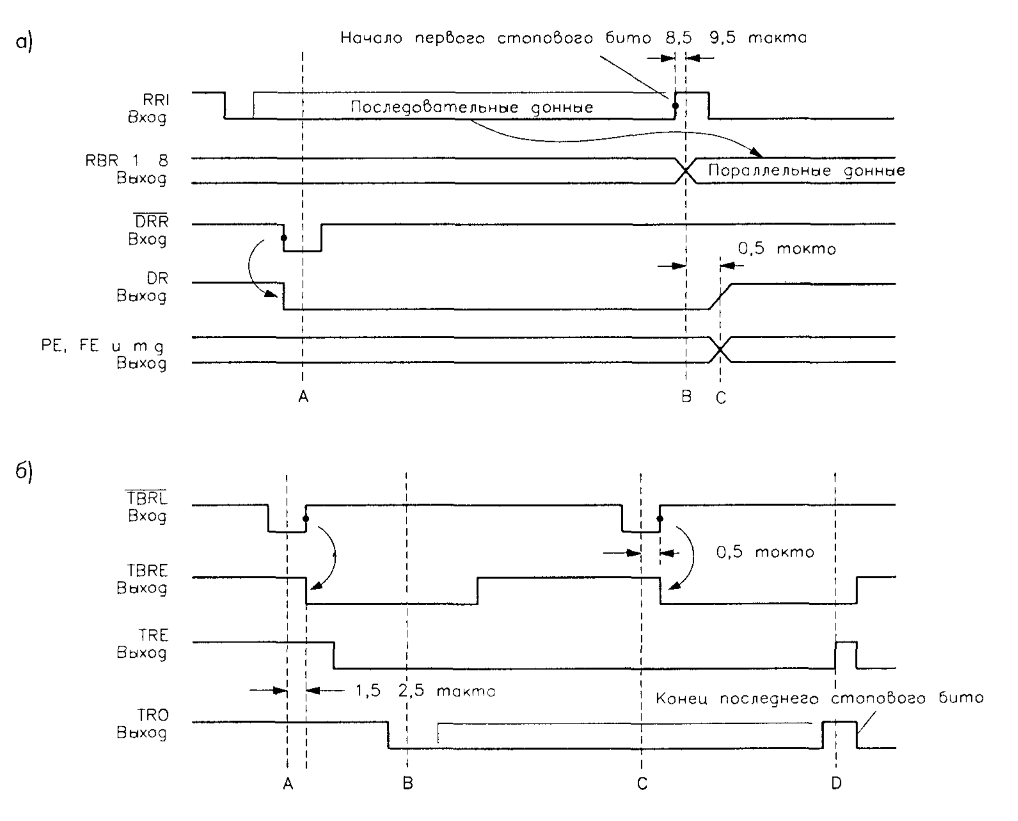


Рис. 4.9. Временные диаграммы: а – приема; б – передачи

Схема с использованием UART CDP6402, соединенная с экспериментальной платой последовательного порта, показана на рис. 4.10. Линии RRI, TRO и $\overline{\text{TBRL}}$ соединены с контактами TD, RD и DTR на экспериментальной плате последовательного порта. Тактовый генератор собран на микросхеме CD4060 и кварце 2,4575 МГц. Тактирующий сигнал частотой 153,6 кГц поступает с контакта 7 микросхемы CD4060. Формат передачи данных следующий: скорость 9600 бод, длина блока данных 8 бит, длина стоповой посылки 1 бит, без проверки на четность. Контакт 18 ($\overline{\text{DRR}}$) соединен с общим проводом. Это означает, что UART принимает данные в непрерывном режиме.

Проводя опыты с указанной схемой, можно использовать программное обеспечение для экспериментальной платы последовательного порта (см. главу 3). Разрешается передавать данные с компьютера на предлагаемое устройство. При изменении состояния линии DTR из единицы в нуль и затем вновь в единицу данные считываются в компьютер. Для организации циклического теста восемь выходов микросхемы CDP6402 нужно соединить с ее входами. В этом случае значения входных и выходных данных должны быть одинаковыми.



Рис. 4.10. Схема с использованием UART CDP6402

4.2.3. Микросхема ITC232-A для сопряжения с последовательным портом

ITC232-A (Timely Technology, RS213-7312) – это новая интерфейсная интегральная схема, разработанная для упрощения сопряжения с последовательным портом ПК при помощи трех линий: TD, RD и общего провода (рис. 4.11). Микросхема характеризуется мощным встроенным набором команд управления и транслятором машинного кода. Команду можно вводить с клавиатуры компьютера – ИС преобразует ее в машинный код и выполнит соответствующие действия. К числу преимуществ ITC232-A относится то, что для работы с ней не нужно учить сложные языки низкого уровня и аппаратное управление; кроме того, нет необходимости компилировать команды.

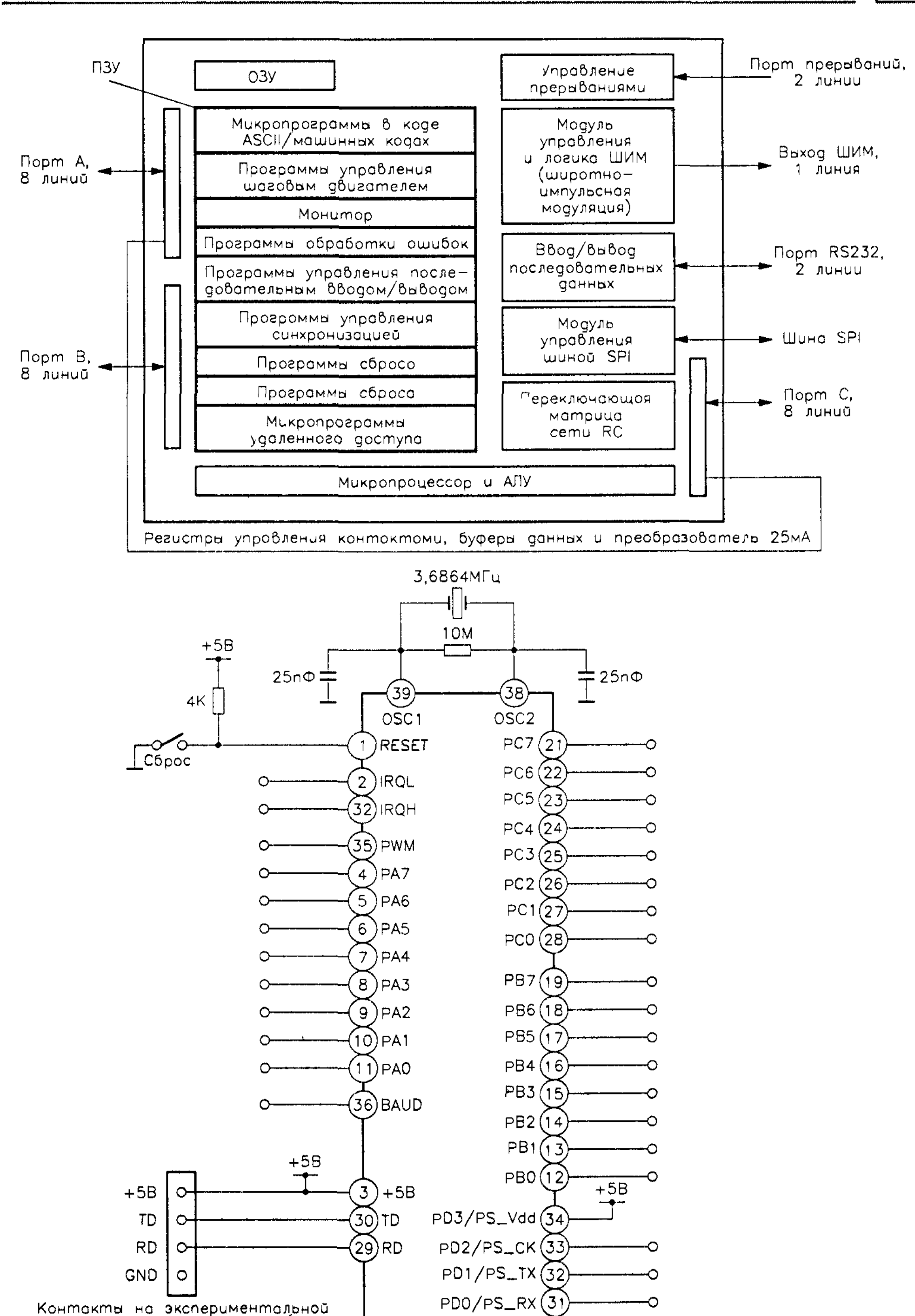


Рис. 4.11. Внутренняя блок-схема и типовое включение ITC232-A

Микросхема имеет 40-контактный двухрядный корпус типа DIP. Для нее требуются источник питания +5 В, потребляемый ток 50 мА. Последовательный порт может работать со скоростями 300–115200 бод.

24 линии ввода/вывода разделены на три порта: A, B и C. Порты могут быть независимо настроены как входы или выходы. Биты 4–7 всех портов способны управлять двухфазными шаговыми двигателями. Скорость вращения варьируется от 10 до 4000 об/с. Биты 0–3 можно использовать для измерения сопротивления или емкости. Устройство выдает сигнал ШИМ (широтно-импульсная модуляция) частотой 10–10000 Гц и скважностью 1–100%. Микросхема также оснащена шиной SPI, с помощью которой к ней подключается любое SPI-совместимое устройство, например 10-канальный АЦП MC145041.

ITC232-A соединяется с экспериментальной платой последовательного порта, как показано на рис. 4.11. Если с компьютера послать команду PWA254, то в порт запишется число 254. При команде SAL100 шаговый двигатель, подключенный к порту A, будет вращаться влево со скоростью 100 об/с. Команда W1000 заставит устройство генерировать на выходе PWM тон частотой 1 кГц. Команды, посылаемые в контроллер, можно набирать в командной строке или включить в самостоятельно написанную программу.

4.3. Увеличение количества линий игрового порта

У игрового порта есть два (или четыре) цифровых и два (или четыре) аналоговых входа и нет выходов. На цифровые входы может подаваться как последовательная, так и параллельная цифровая информация. Если данные последовательные, то перевести их в параллельный формат можно с помощью программного обеспечения компьютера.

4.4. Последовательно-параллельные преобразователи

Последовательно-параллельные регистры сдвига дают возможность при наличии всего двух входных линий получить неограниченное количество выходных. На рис. 4.12а изображена схема на основе ИС 74LS164, предназначенная для получения восьми выходных линий. 74LS164 имеет два входа с последовательной загрузкой, контакты 1 и 2 (A и B) и восемь выходных регистров сдвига (Qa – Qh). Данные на входах A и B по положительному фронту входного тактирующего сигнала сдвигаются в ячейку Qa, затем перемещаются в Qb, Qc и т.д. После восьми тактовых импульсов на выходах регистра сдвига присутствует 8 бит данных, причем первый бит записан в ячейку Qh. При подаче на контакт 9 ($\overline{\text{CLEAR}}$) сигнала низкого уровня на всех восьми выходах появляется ноль. Максимальная частота тактового сигнала 25 МГц. Для получения большего количества выходов можно последовательно объединить несколько таких микросхем.

На рис. 4.12а показано соединение описанной схемы с экспериментальной платой последовательного порта. Линия RTS подключена к последовательному входу

Рис. 4.12. Схемы последовательно-параллельных преобразователей. а – на основе 74LS164; б – на основе 74LS164 и 74LS374

(контакты 1 и 2), а линия DTR – к тактирующему входу (контакт 8). При загрузке данных они сначала передаются по линии RTS, а затем на вход DTR поступает отрицательный импульс. При проведении экспериментов со схемой можно использовать программное обеспечение для экспериментальной платы последовательного порта (см. главу 3).

Последовательно-параллельное преобразование имеет два недостатка. Во-первых, компьютеры Pentium генерируют тактовый сигнал частотой 0,1–1 МГц. Период загрузки восьми бит данных можно вычислить. Таким образом, чем больше выходов в схеме, тем меньше скорость передачи данных; это не критично для низко- и среднескоростных приложений сопряжения. Во-вторых, во время загрузки данных содержимое регистров сдвига изменяется случайным образом. Для решения этой проблемы применяются триггеры-защелки, в частности 74LS374 (рис. 4.12б). После загрузки данных во все регистры они передаются в 74LS374 по положительному фронту тактирующего импульса (контакт 11 ИС). Однако схема требует наличия дополнительного выхода из компьютера (в данном примере используется линия TD).

Приведенные схемы можно подключать и к экспериментальной плате параллельного порта. При этом три линии регистра данных или регистра управления должны функционировать как выходы.

4.5. Параллельно-последовательные преобразователи

С помощью *параллельно-последовательных регистров сдвига* типа CD4021 увеличивается количество входов. Для этого требуются две выходные линии компьютера (для загрузки параллельных данных и для сдвига данных) и одна входная линия для считывания информации. На рис. 4.13 изображена схема для ввода восьми бит данных, а также указано назначение и расположение выводов.

Микросхема имеет тактирующий вход (контакт 10), вход управления параллельным/последовательным вводом ($P/\overline{S}$, контакт 9), вход/выход последовательных данных (контакт 11), восемь параллельных входов (D0 – D7) и три последовательных выхода (Q6 – Q8). В начале работы на входы D0 – D7 необходимо подать информацию. Затем по положительному фронту на входе $P/\overline{S}$ данные параллельно записываются во внутренний регистр независимо от состояния тактового импульса.

Отрицательный фронт на входе $P/\overline{S}$ инициирует операцию последовательного вывода. По положительному фронту тактового импульса данные передаются с выхода Q7. После восьми тактовых импульсов все входные данные будут переданы.

Работу микросхемы совместно с экспериментальной платой последовательного порта иллюстрирует рис. 4.13. Тактирующий вход соединяется с линией DTR, вход $P/\overline{S}$ – с RTS, выход Q8 – с CTS. При проведении опытов можно воспользоваться программным обеспечением для экспериментальной платы последовательного порта (см. главу 3). ИС также может подключаться к параллельному порту. Скорость передачи информации невысока, поэтому CD4021 применяется только для средне- и низкоскоростных приложений сопряжения.

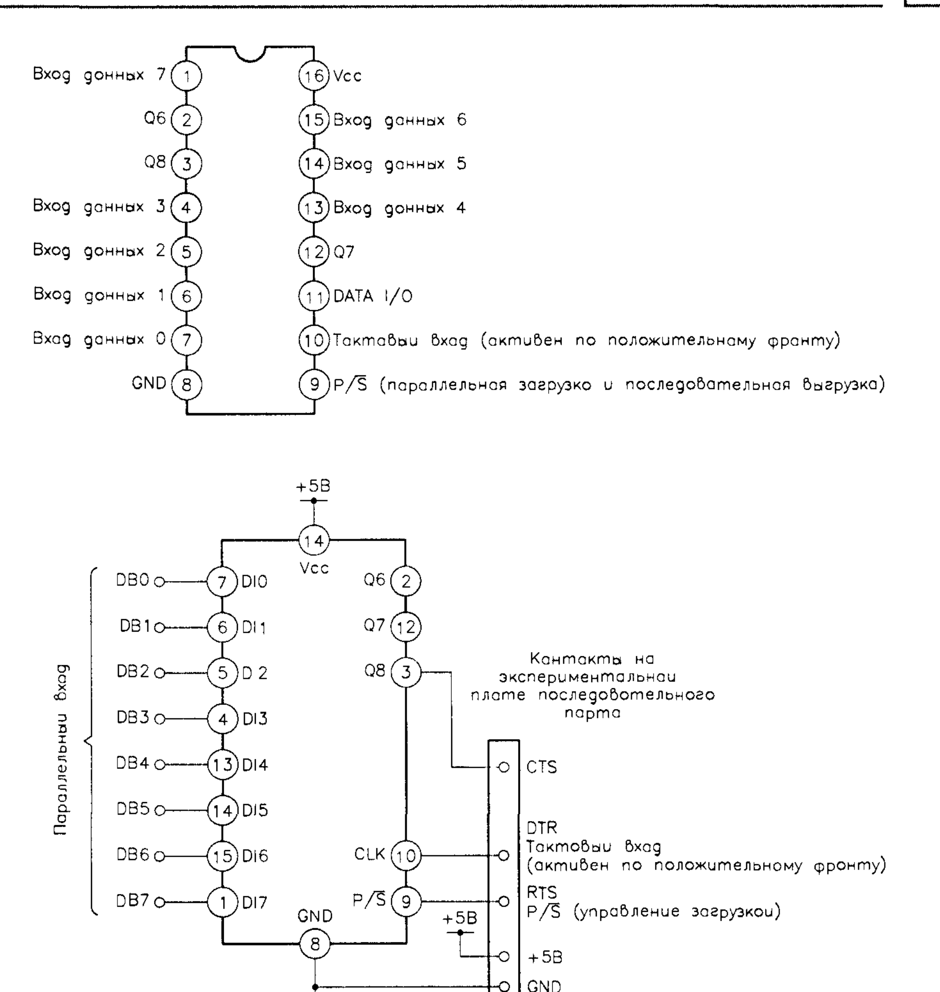


Рис. 4.13. Параллельно-последовательный преобразователь на базе микросхемы CD4021

4.6. Шифраторы и дешифраторы данных

Еще один способ увеличения количества цифровых линий ввода/вывода – использование шифраторов и дешифраторов – микросхем со средней степенью интеграции, специально разработанных для приложений цифровой связи и передачи данных, в частности ИС серии MC1405XX (National Semiconductors) или семейства HT-12 (Holtec) Шифраторы представляют собой параллельно-последовательные преобразователи, а дешифраторы – последовательно-параллельные Соединение между шифратором и дешифратором можно установить как с помощью кабельных линий связи, так и другими способами, например посредством инфракрасных, волоконно-оптических, ультразвуковых или радиолиний Скорость передачи данных при этом достаточно мала

Микросхема HT-12E (рис. 4.14) передает последовательно перекодированные данные при подаче сигнала низкого уровня на вход 14 (передача разрешена, $\overline{TE}$). Входные данные представляют собой 12 бит – это 8 бит адреса (A0 – A7, контакты 1–8) и 4 бита данных (D0 – D3, контакты 10–13). Общее количество комбинаций адреса равно 256. Между контактами 15 и 16 включается внешний резистор (допуск 5%), Выбирая значения сопротивления этого резистора, можно получать разные скорости передачи данных. Резистор фактически задает внутреннюю тактовую частоту, на которой работает микросхема. Последовательные данные выводятся через контакт 17. Микросхема функционирует в широком диапазоне напряжений (2,4–12 В); ток потребления в режиме покоя 1 мкА.

Исходное состояние микросхемы – режим ожидания. При подаче сигнала низкого уровня на вход $\overline{TE}$ начинается цикл передачи по четыре слова и продолжается

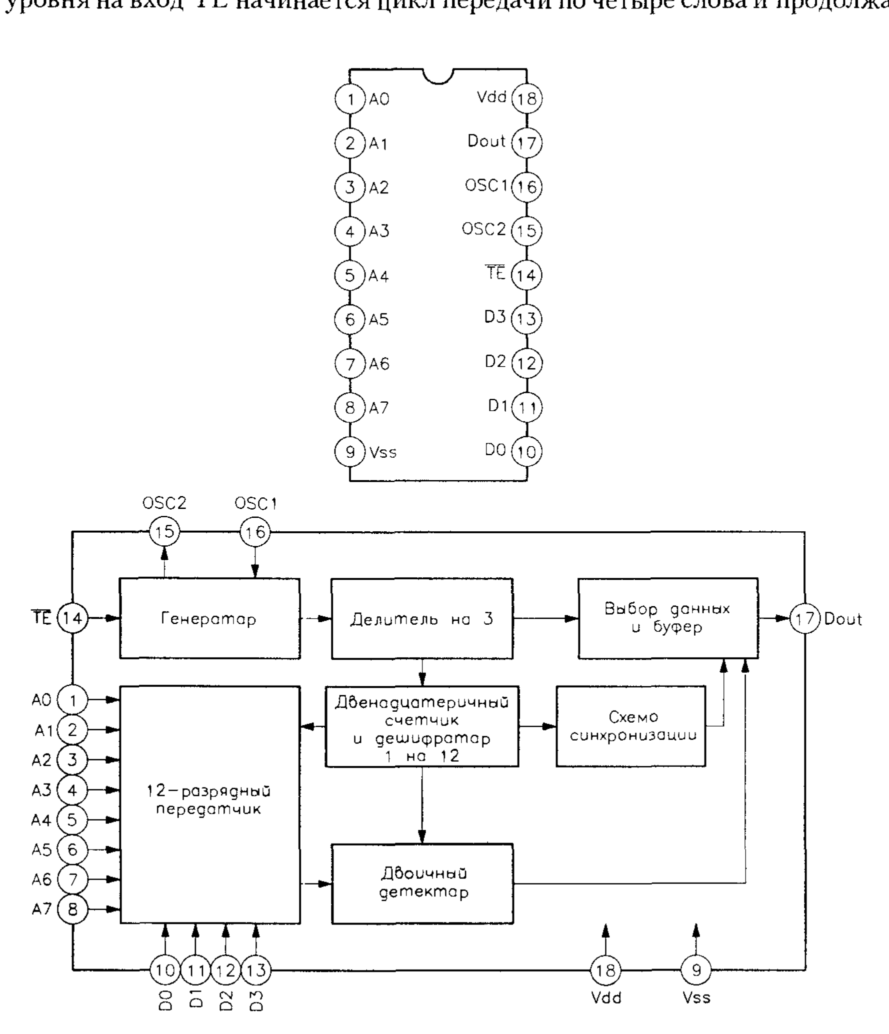


Рис. 4.14. Расположение выводов и внутренняя блок-схема шифратора HT-12E

до тех пор, пока на $\overline{TE}$ не появится единица. В каждом слове содержатся пилот-сигнал и код (рис. 4.15а). Пилот-сигнал имеет длительность 12 бит и нулевой логический уровень, код также состоит из 12 бит. Микросхема считывает состояния 12 входов (A0 – A7, D0 – D3) и передает эту информацию. Логические 0 и 1 кодируются, как показано на рис. 4.15б.

Дешифратор HT-12D (рис. 4.16) принимает 12 бит последовательных данных, распознает первые восемь бит как адрес и последние четыре бита как данные. Если принятый адрес совпадает с заранее установленным, то данные сохраняются во внутреннем регистре. Это условие проверяется трижды. Если же полученный адрес отличается от заранее установленного, то процесс приема прерывается, и микросхема сбрасывается в исходное состояние. Предустановка адреса определяется логическим состоянием выводов 1–8. Принятые четыре бита данных снимаются с выводов 10–13. Последовательные входные данные поступают на вход 14. Внешний резистор (допуск 5%) подключается между контактами 15 и 16. Контакт 17 – это выход индикатора правильной передачи.

Здесь также необходим резистор с допуском 5%, который подключается между контактами 15 и 16 и определяет скорость работы микросхемы. Внутренние тактовые частоты HT-12E и HT-12D связаны между собой следующим соотношением:

$$F_{OSC,\ HT\text{-}12D} = 50F_{OSC\ HT\text{-}12E}\ .$$

Например, если тактовая частота шифратора равна 3 кГц, то тактовая частота дешифратора составляет 150 кГц. В этом случае внешний резистор для шифратора должен иметь сопротивление 1,1 МОм, а для дешифратора – 62 кОм.

Следующая программа на языке TP6 эмулирует работу компьютера в качестве микросхемы HT-12E и выдает двенадцатибитовые последовательно перекодированные данные. Информация подается на вход дешифратора HT-12D через контакт DTR на экспериментальной плате последовательного порта с тактовой частотой

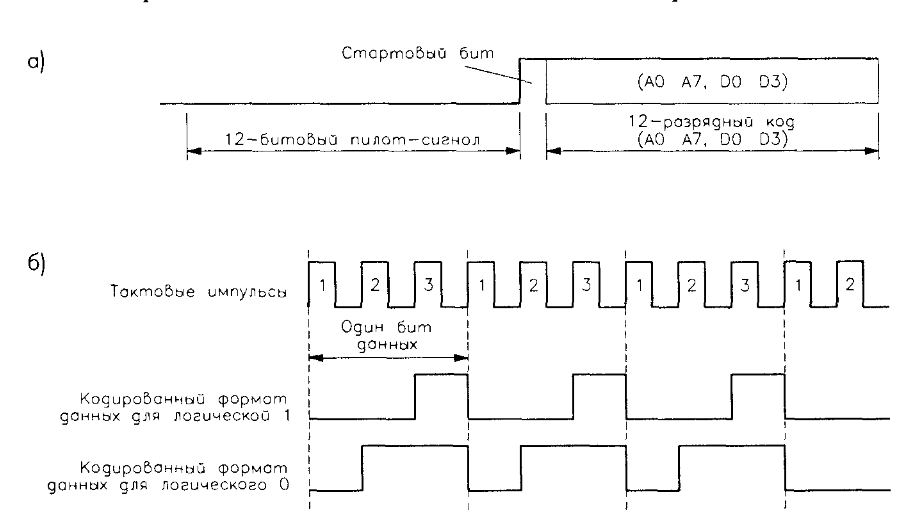


Рис. 4.15. Формат последовательно перекодированных данных: а – при передаче одной посылки, б – для логических единицы и нуля

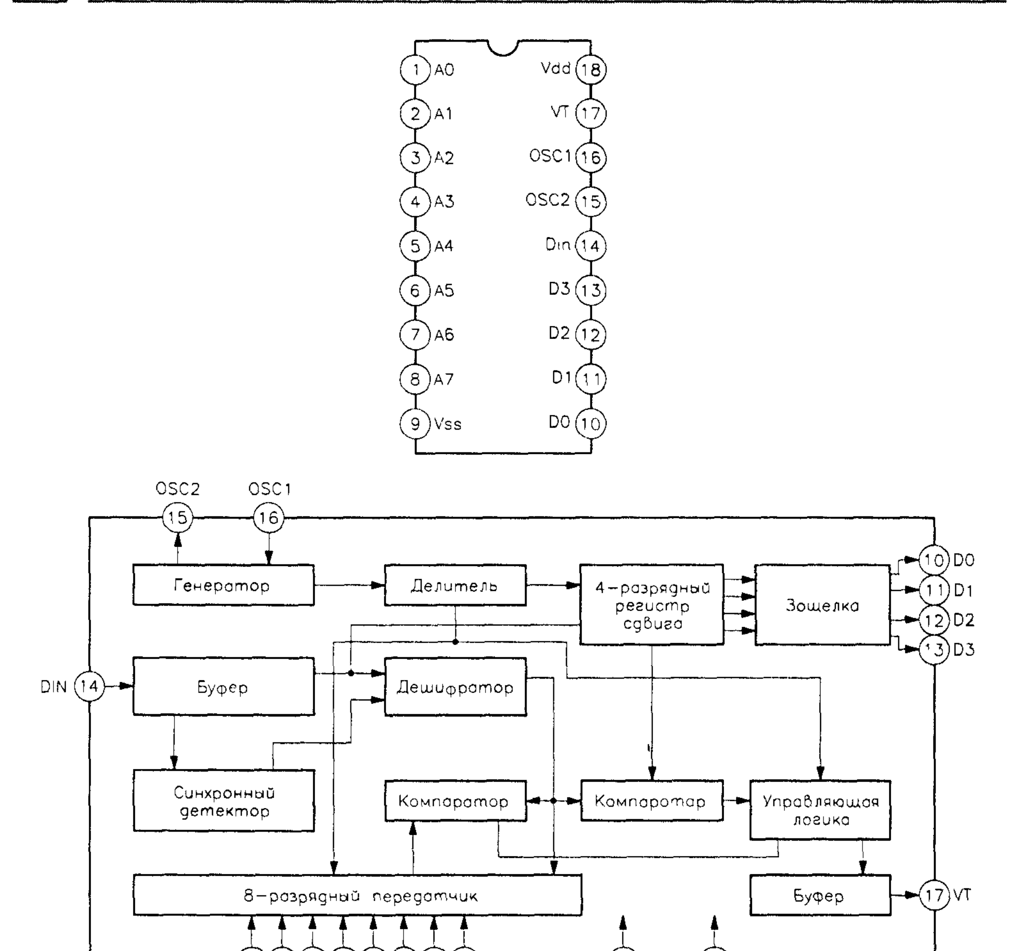


Рис. 4.16. Внутренняя блок-схема дешифратора HT-12D

1 кГц. Внешний резистор имеет номинал 220 кОм[1], что дает системную тактовую частоту 50 кГц. Вход данных (контакт 14) соединен с линией DTR на экспериментальной плате последовательного порта (рис. 4.17).

Текст программы HT_12E.PAS

```
Program HT_12_encoder,
(*Программа для эмуляции сигнала в формате микросхемы HT-12E.*)
uses
```

[1] Согласно фирменной документации, для получения тактовой частоты 50 кГц сопротивление резистора должно равняться 200 кОм. – *Прим. науч. ред.*

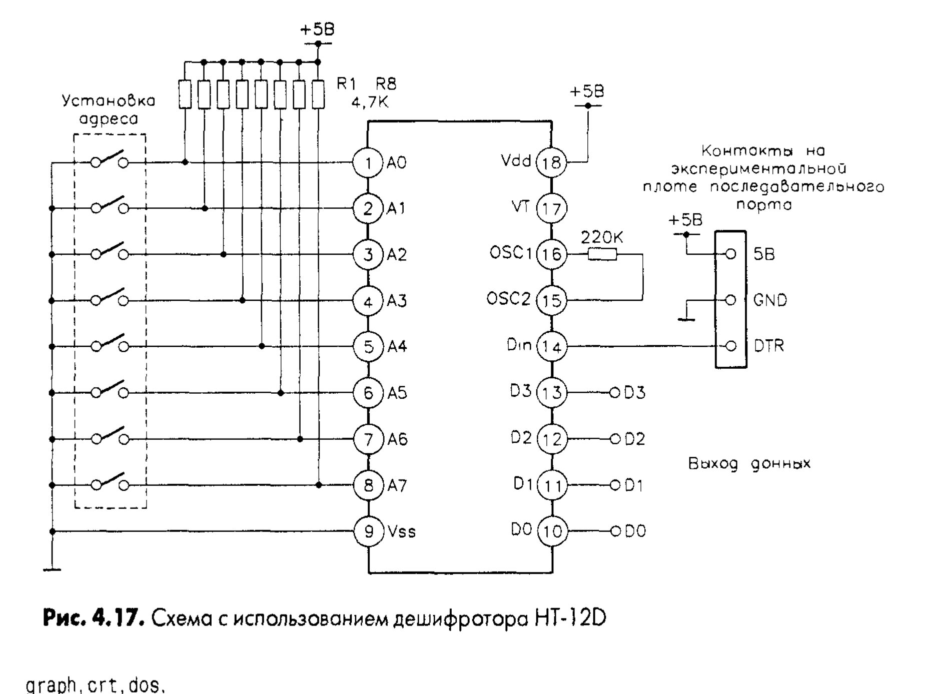


Рис. 4.17. Схема с использованием дешифратора HT-12D

```
  graph,crt,dos.
var
  addressx,datax integer;
  bit array[1  15] of byte;
  i,serial_input_byte.baud_rate_byte,data_length_byte,stop_length_byte,parity_byte integer,
  ch char;

(*Загрузка файла библиотеки *)
{$I c:\ioexp\tplib1 pas}

Procedure
Transmit_serial_data(Port_address,bit,original_data,address,data:integer,invert_flag.Boolean),
(*Передача данных в формате HT-12E. частота передатчика 1000 Гц.*)
(*Port_address: адрес порта ввода/вывода,
Bit· используется для передачи данных,
Original_data· исходные данные порта,
Address  передаваемые 8 бит данных,
Data: передаваемые 4 бита данных,
Invert_flag=1, при этом передаваемые данные инвертируются.*)
Var
  i,transmit_time.byte;
  data_bit array[1  73] of byte;
begin
(*Формат данных HT-12E·
сначала 36 тактовых интервалов пилот-сигнала,
1/2 тактового интервала - стартовая посылка,
24 тактовых интервала - адрес (A0 - A7)
```

```
и 12 тактовых интервалов - данные (D0 - D3).*)

(*Присвоение значения переменной Data_bit.*)

(*Пилот-сигнал.*)
  for i:=1 to 36 do if not invert_flag then data_bit[i]:=original_data and (255-bit_weight(bit))
  else data_bit[i]:=original_data or bit_weight(bit);

  (*Стартовая посылка.*)
  if not invert_flag then original_data or bit_weight(bit)
  else data_bit[i]:=original_data and (255-bit_weight(bit));

  (*Биты адреса. A1 передается первым.*)
  for i:=1 to 8 do
     begin
        if not invert_flag then
           begin
              data_bit[3*(i-1)+37+1]:=original_data and(255-bit_weight(bit));
              data_bit[3*(i-1)+37+2]:=original_data or bit_weight(bit);
              data_bit[3*(i-1)+37+3]:=original_data or bit_weight(bit);
           end
        else
           begin
              data_bit[3*(i-1)+37+1]:=original_data or bit_weight(bit);
              data_bit[3*(i-1)+37+2]:= original_data and(255-bit_weight(bit));
              data_bit[3*(i-1)+37+3]:=original_data and(255-bit_weight(bit));
           end;
        if address and bit_weight(i)>0 then
           begin
              if not invert_flag then data_bit[3*(i-1)+37+2]:= original_data and(255-
                                                                  bit_weight(bit))
              else
                 data_bit[3*(i-1)+37+2]:=original_data or bit_weight(bit);
           end;
  end;

  (*Биты данных.*)
  for i:=1 to 4 do
     begin
        if not invert_flag then
           begin
              data_bit[3*(i-1)+61+1]:=original_data and(255-bit_weight(bit));
              data_bit[3*(i-1)+61+2]:=original_data or bit_weight(bit);
              data_bit[3*(i-1)+61+3]:=original_data or bit_weight(bit);
           end
        else
           begin
              data_bit[3*(i-1)+61+1]:=original_data or bit_weight(bit);
              data_bit[3*(i-1)+61+2]:= original_data and(255-bit_weight(bit));
              data_bit[3*(i-1)+61+3]:=original_data and(255-bit_weight(bit));
           end;
        if data and bit_weight(i)>0 then
           begin
              if not invert_flag then data_bit[3*(i-1)+61+2]:= original_data and(15-
                                                                  bit_weight(bit))
              else
```

```
                    data_bit[3*(i-1)+37+2] =original_data or bit_weight(bit),
                end,
    end

    (*Десятикратная передача кода *)
    for transmit_time =1 to 10 do
        begin
            for i =1 to 73 do
                begin
                    port(Port_address) =data_bit[i],
                    delay(11),  (*Задержка на 1 мс  Опорная частота 1 кГц *)
                end,
        end,
end,

(*Главная программа *)
begin
    COM_address,
    port(RS232_address) =0,
    Repeat
        clrscr
        write( Input the address of receiver HT-12D (0-255)  ), readln(addressx),
        write( Input the data to be transmitted (0-15)  ), readln(datax),
        transmit_serial_data(RS232_address+4,0 0 addressx,datax,true),
        write( Quit the program [Y/N]  ), readln(ch),
    until upcase(ch)='Y',
end
```

Шифратор HT-12E (рис. 4.18) можно использовать как двенадцатибитовый параллельно-последовательный преобразователь. Следующая программа на языке TP6 считывает последовательные данные, поступающие от шифратора HT-12E в компьютер через линию CTS последовательного порта, трансформирует их в параллельную форму (8 бит адреса и 4 бита данных) и выводит на экран.

Внешнее сопротивление равно 2,5 МОм, что формирует тактовую частоту микросхемы 1 кГц. Контакт 17 (Dout) соединен с контактом CTS на экспериментальной плате последовательного порта. Вместо последовательного порта в рассматриваемом примере можно использовать игровой. При этом Dout необходимо соединить с одним из цифровых входов порта.

Текст программы HT_12D.PAS

```
Program HT_12_receiver,
(*Программа эмуляции дешифраторов HT-12D *)
uses
    graph,crt,dos,
var
    value,oldvalue integer,
    bit array[1  15] of byte
    i,led_selected,serial_input_byte,baud_rate_byte,data_length_byte,
stop_length_byte,parity_byte integer,

(*Загрузка файла библиотеки *)
{$I c \ioexp\tplib1 pas}
```

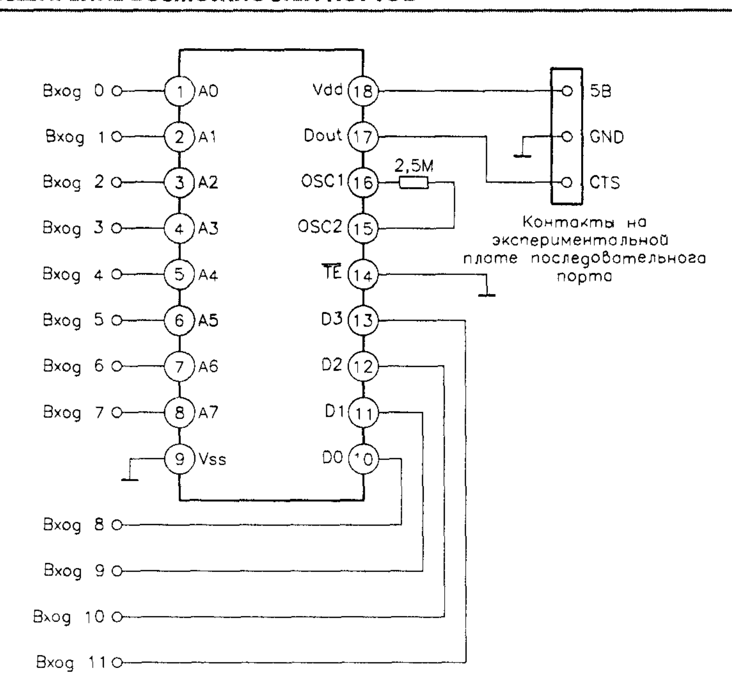


Рис. 4.18. Схема с использованием шифратора HT-12E

```
function serial_data(x byte) byte,
(*Декодирование сигнала HT-12E  Вход соединен с линией CTS *)
(*X=1 для A0-A7, X=2 для D0-D3 *)
var
  sdata, saddress, i byte,
  timeI_0, timeI_1, clockI, timeI longint,
begin
  (*Нахождение длительности посылки высокого уровня (программно) *)
  repeat delay(1) until port(RS232_address+6) and 16=16, (*Определение низкого уровня
                                                          сигнала *)
  repeat delay(1) until port(RS232_address+6) and 16=0  (*Определение перехода из нуля
                                                          в единицу *)
  ClockI =0,
  Repeat
     ClockI =clockI+1,
     delay(1)
  until port(RS232_address+6) and 16=16,

  (*Определение пилот-сигнала как длительного сигнала низкого уровня *)
  repeat
     repeat delay(1) until port(RS232_address+6) and 16=16
     timeI_0 =0
     repeat
        timeI_0 =timeI_0+1
```

```
            delay(1);
        until port(RS232_address+6) and 16=0;
        until timeI_0>12*clockI;
        clockI:=0;  (*Определение длительности стартового бита.*)
    repeat delay(1) until port(RS232_address) and 16=0;
    repeat
        clockI:=clockI+1;
        delay(1),
    until port(RS232_address+6) and 16=16;    (*Определение перехода из единицы в нуль.*)
    (*Считывание следующих 12 бит.*)
    for i:=1 to 12 do
        begin
            timeI_1:=0;
            repeat delay(1) until port(RS232_address+6) and 16=0;
            repeat
                timeI_1:=timeI_1+1;
                delay(1);
            until port(RS232_address+6) and 16=16;
            if abs(timeI_1-clockI)>clockI/2 then bit[i]:=0 else bit[i]:=1;
        end;
    sdata:=0;
    saddress:=0;
    for i:= 1 to 8 do saddress:=saddress+bit[i]*bit_weight(i),
    for i:=9 to 12 do sdata:=sdata+bit[i]*bit_weight(i-8),
    if x=1 then serial_data:=saddress,
    if x=2 then serial_data:=sdata;
    end,

    (*Главная программа *)
    begin
        COM_address;
        repeat
            write('Received serial address=',serial_data(1));
            delay(300),
            write('Received serial data=',serial_data(2)),
            readln;
    until keypressed;
end.
```

4.7. Шина I²C

I²C – это шина данных, созданная фирмой Philips для связи между интегральными схемами или модулями. Она позволяет устройствам обмениваться данными посредством только двух линий, что намного упрощает разработку сложных электрических схем. Существует целое семейство I²C-совместимых устройств для различных приложений: увеличения количества линий ввода/вывода, цифро-аналоговых и аналого-цифровых преобразований, отсчета времени, модулей памяти, синтеза частот и т.д.

Введя I²C в компьютер, вы убедитесь в хорошей производительности самой шины и микросхем поддержки. Для реализации шины I²C применяются как параллельный, так и последовательный порты.

4.7.1. Принцип работы

Шина состоит всего из двух линий: двунаправленной линии данных (SDA) и линии импульсов тактов (SCL), которые соединяются с положительным напряжением питания через нагрузочные резисторы (рис. 4.19).

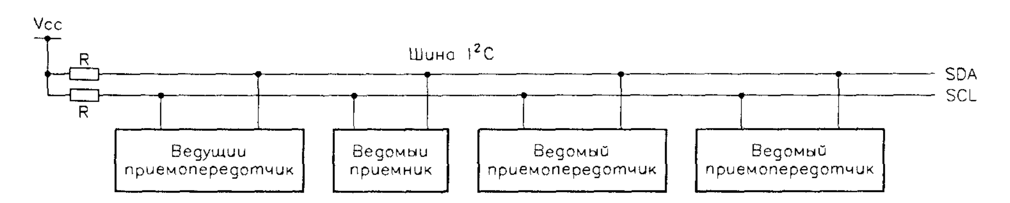


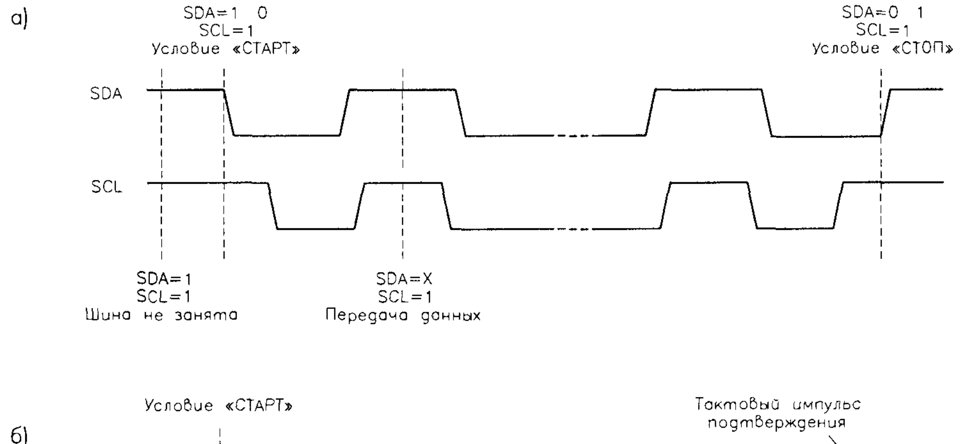
Рис. 4.19. Функциональная схема шины I²C

Передатчик генерирует сообщение, приемник его получает. Ведущее устройство контролирует работу шины. Для шины I²C определен следующий протокол соединений:

- передачу данных можно начать только в том случае, если шина не занята;
- во время передачи состояние линии данных должно оставаться постоянным при сигнале высокого уровня на линии тактовых импульсов.

Изменение состояния линии данных при единице на тактирующей линии рассматривается как передача сигналов управления. Определены следующие корректные состояния во время работы шины (см. рис. 4.20):

- *шина не занята*: на обеих линиях единица;
- *начало передачи данных*: изменение состояния линии данных из единицы в ноль, тогда как линия тактов остается в единичном состоянии и определяет условие начала передачи (условие «СТАРТ»);
- *прекращение передачи данных*: изменение состояния линии данных из нуля в единицу, при этом линия тактов остается в единичном состоянии и определяет условие окончания передачи (условие «СТОП»);
- *корректность данных*: любое состояние линии данных после старта остается корректным и неизменным, если на линии тактов единица. Данные можно менять, если на линии тактов низкий уровень. На один бит информации приходится один тактовый интервал. Каждый цикл передачи данных начинается условием «СТАРТ» и заканчивается условием «СТОП». Количество битов данных, передаваемых между этими состояниями, не ограничено. Данные передаются побайтно; приемник удостоверяет получение, пересылая бит подтверждения после приема каждого байта;
- *бит подтверждения*: передается после каждого байта. Активный передатчик формирует сигнал высокого уровня на шине SDA, и ведущий передатчик в этот момент генерирует дополнительное подтверждение высоким уровнем тактового импульса по шине SCL. Приемник выдает сигнал подтверждения



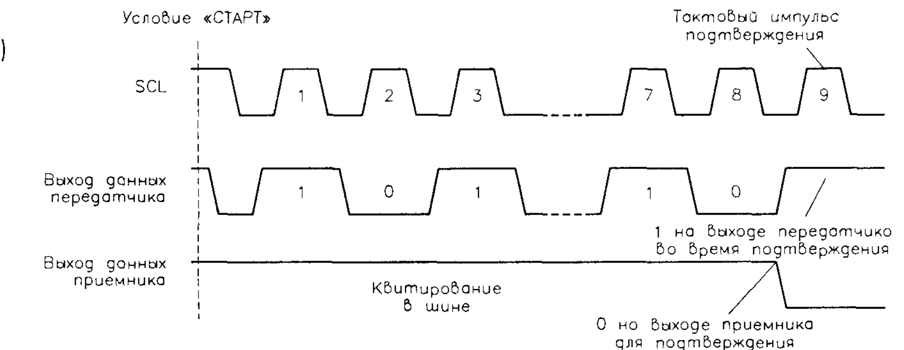


Рис. 4.20. Принцип работы шины. а – состояния, б – квитирование

низким уровнем. Приемник, который получает данные, генерирует бит подтверждения после приема каждого байта.

Приемник, который генерирует бит квитанции, переключает линию SDA в низкий уровень и удерживает ее в этом состоянии до тех пор, пока тактовый импульс подтверждения находится в состоянии высокого уровня. Для прекращения передачи данных приемник должен оставить принятый байт без подтверждения.

4.7.2. Временные диаграммы работы шины I²C

Перед тем как передать данные по шине, необходимо получить адрес отвечающего устройства с помощью передачи 7 бит адреса и бита $R/\overline{W}$ (низким уровнем) после стартовой посылки. Типовой формат байта адреса следующий:

Биты постоянного адреса + биты программируемого адреса + бит $R/\overline{W}$.

Постоянный адрес зависит от самой микросхемы и не изменяется. Программируемый адрес можно задать посредством адресных выводов микросхемы. Последний бит – бит чтения/записи – указывает направление потока данных. Следом за байтом адреса передается байт управления, который также определяется



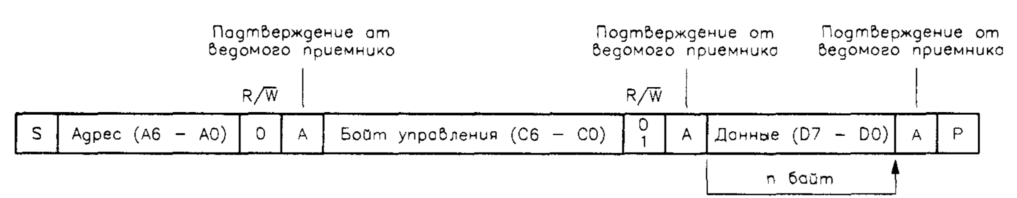


Рис. 4.21. Формат последовательной передачи данных шины I²C

используемой микросхемой. Затем пересылаются байты данных. Формат последовательных данных изображен на рис. 4.21.

4.7.3. Реализация на базе параллельного и последовательного портов

Экспериментальная схема простой реализации шины I²C на базе параллельного порта изображена на рис. 4.22. Данная схема работает по стандарту I²C, при этом компьютер выступает в качестве ведущего устройства. Два транзистора обеспечивают

Рис. 4.22. Реализация шины на базе параллельного и последовательного портов

выходы линий данных и тактов. Напряжение шины +5 В. Здесь можно использовать любой n–p–n транзистор с частотой 100 кГц и повторители с открытым коллектором 7407. Линия SCL соединена с контактом D2 на экспериментальной плате параллельного порта (бит 1 регистра данных). Линия SDA подключена к контакту D1 (бит 0 регистра данных) и считывается в компьютер через контакт S1 на экспериментальной плате (бит 3 регистра состояния).

Экспериментальная схема реализации шины на базе последовательного порта также показана на рис. 4.22. Линиями SCL и SDA управляют соответственно контакты DTR и RTS. Линия SDA считывается в компьютер через контакт CTS. Примеры программного обеспечения для таких схем приведены в главе 7.

4.7.4. Микросхемы, поддерживающие стандарт I^2C

Некоторые примеры I^2C-совместимых ИС приведены ниже:

- драйверы жидкокристаллических дисплеев: PCF8466, PCF8576, PCF8577, PCF8578/79, SAA1064;
- модули памяти: PCF8570, PCF8572/8573, PCF8582A, PCF8598-2T;
- устройства отсчета времени: PCF8583, 41T56C, PCF8593, PCF8598;
- интерфейсы ввода/вывода: PCF8574, PCF8584, PCF8582, PCF82B715;
- восьмиразрядные четырехканальные АЦП и ЦАП: PCF8591;
- инфракрасные передатчики: SAA3028;
- DTMF-кодеки: PCD3311/3312;
- синтезаторы речи: PCF8200;
- аудиоинтерфейс: SAA1136.

4.8. Последовательный периферийный интерфейс

Шина SPI позволяет соединять внешние устройства с главным компьютером при помощи трех линий. Первая линия тактирующая и представляет собой вход микросхемы, по второй линии данные принимаются, по третьей – передаются. К одной шине можно подключить несколько SPI-совместимых устройств. Если микросхема соединяется с компьютером, то на ее контакт «выбор чипа» ($\overline{CS}$) необходимо подать сигнал низкого уровня. Шина SPI имеется в микросхемах различных серий, включая АЦП, ЦАП, аналоговые мультиплексоры и т.п. Ее можно реализовать как на базе параллельного, так и последовательного портов. Сделав это, вы существенно упростите сопряжение ПК с различными микросхемами.

4.9. Шина MicroLAN

Шина MicroLAN разработана фирмой Dallas Semiconductors и использует только одну линию и заземление. Она имеет единственное ведущее устройство и множество ведомых с 2^{56} логическими адресами. Максимальная длина кабеля (дешевый неэкранированный телефонный кабель) более 300 м без ретрансляции сигнала. Шина позволяет соединять различные устройства: блоки энергонезависимой памяти, температурные датчики, системы отсчета времени, всевозможные коммутаторы

и т.п., каждое из которых обладает уникальным сетевым идентификационным номером. Шина имеет стандартный уровень напряжений ТТЛ/КМОП, а все устройства в сети работают от питания по линии данных. Напряжение 0–0,8 В определяет уровень логического нуля, а напряжение выше 2,2 В – логической единицы. Для работы всех устройств необходимо напряжение 2,8 В. Скорость передачи данных достигает 16300 бит/с. Шина может быть реализована на базе последовательного порта компьютера, при этом ПК выступает в роли ведущего устройства, все остальные устройства – ведомые.

4.10. Сопряжение между схемами ТТЛ и КМОП

Часто требуется соединить выход логических схем ТТЛ со входом КМОП и наоборот. На рис. 4.23 показано несколько способов такого сопряжения.

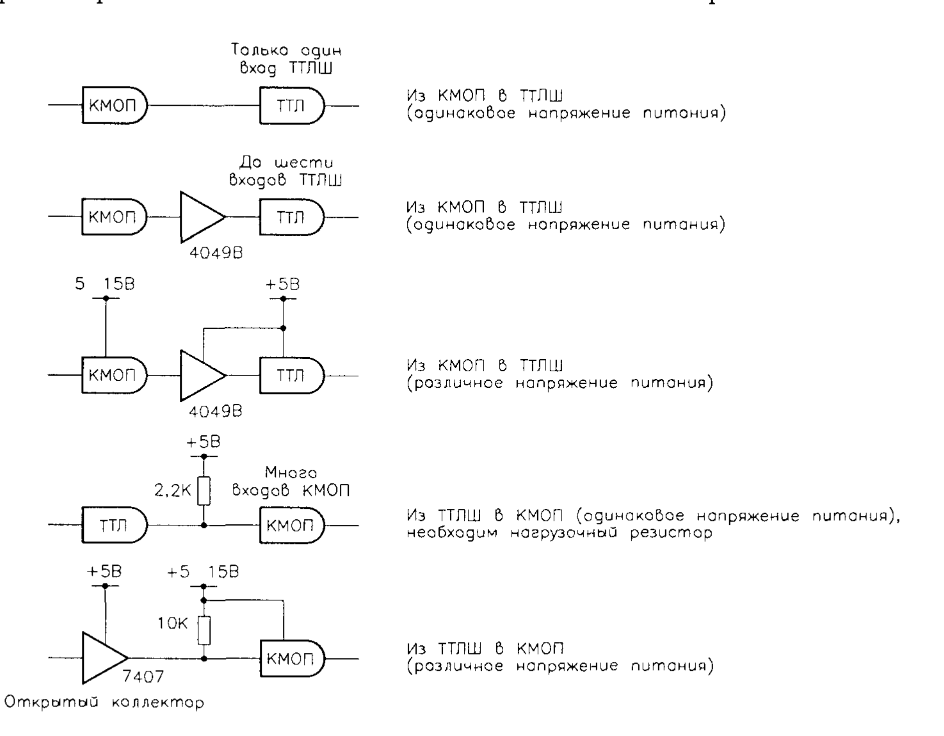


Рис. 4.23. Сопряжение схем КМОП и ТТЛ

Если в схемах ТТЛ и КМОП одинаковое напряжение питания, то схема КМОП может непосредственно управлять схемой ТТЛШ без дополнительной обвязки. Если же необходимо управлять сразу несколькими схемами ТТЛШ, то для усиления тока допустимо использовать КМОП буфер типа 4049В. Когда схема ТТЛШ управляет схемой КМОП, нужен нагрузочный резистор 2,2 кОм.

4.11. Защита цифровых линий ввода/вывода

Наиболее простой способ защиты линий ввода/вывода заключается в применении схем стабилизации напряжения на базе резисторов и стабилитронов. Однако чаще всего используются оптоэлектронные компоненты. Типовая микросхема состоит из инфракрасного светодиода и фототранзистора или фотодиода, которые образуют оптопару. Разность потенциалов между входом и выходом подобных микросхем может составлять порядка нескольких киловольт. Типовые схемы включения приведены на рис. 4.24.

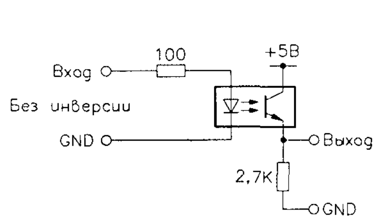


Рис. 4.24. Приложения, использующие оптоэлектронные компоненты

Существуют различные типы таких ИС. Они классифицируются по электрическим характеристикам входов и выходов и количеству оптопар. На входе могут устанавливаться светодиоды различной мощности. На выходе – транзистор, пара транзисторов, соединенных по схеме Дарлингтона (оптопары Дарлингтона), микросхема КМОП/ТТЛ или триггер Шмитта. Количество встроенных оптопар равно 1, 2 или 4.

Важный параметр микросхем – *коэффициент передачи*, который определяется как отношение выходного тока к входному, выраженное в процентах. Коэффициент передачи 100% означает, что выходной ток равен 1 мА при входном токе 1 мА. Транзисторные оптопары имеют коэффициент передачи 20%, оптопары Дарлингтона – 500%.

PC817, PC827 и PC847 (Sharp, RS175-110, RS175-126, RS175-132) – это одно-, двух- и четырехканальные оптопары с аналогичными характеристиками (рис. 4.25а). Максимальное напряжение изоляции равно 5 кВ (среднеквадратическое значение), коэффициент передачи от 50 до 600%. Номинальное прямое напряжение на светодиоде составляет 1,2 В, а максимальный прямой ток – 50 мА. Максимальное напряжение между коллектором и эмиттером транзистора может достигать 35 В, а максимальный прямой ток транзистора – 50 мА. Время переходных процессов обычно равно 4 мкс. Максимальная частота – 150 кГц.

PC815, PC825 и PC845 (Sharp, RS175-198, RS175-205, RS175-211) – это одно-, двух- и четырехканальные оптопары Дарлингтона (рис. 4.25б). Напряжение изоляции равно 5 кВ (среднеквадратическое значение), коэффициент передачи составляет 600–7500%. Номинальное прямое напряжение светодиода равно 1,2 В, а максимальный прямой ток – 50 мА. Максимальное напряжение между коллектором и эмиттером транзистора Дарлингтона достигает 35 В, а максимальный прямой ток коллектора – 80 мА. Время насыщения 60 мкс, время рассасывания 53 мкс.

Микросхема HCPL-2630 (Toshiba, RS768-116) – это двухканальная высокоскоростная ТТЛ совместимая оптопара (рис. 4.26а,б). Изоляция выдерживает напряжение 2500 В (среднеквадратическое значение). Прямое напряжение на светодиоде

Рис. 4.25. Оптопары: а – транзисторные оптопары PC817, PC827 и PC847; б – оптопары Дарлингтона PC815, PC825 и PC845

составляет 1,5 В, а максимальный прямой ток – 15 мА. Микросхема требует напряжения питания +5 В. Выходной транзистор имеет максимальный ток коллектора 50 мА. Время переходных процессов 30 нс. Максимальная скорость передачи данных 10 Мб/с. Коэффициент разветвления по выходу равен 5.

Оптопары серии 74OL6000 (Quality Technologies) предназначены для работы с ТТЛ и КМОП логическими схемами (рис. 4.26в,г). Они имеют скорость передачи данных 15 Мб/с и напряжение изоляции 2500 В. Входные и выходные напряжения совместимы с ТТЛ/КМОП логикой. Коэффициент разветвления по выходу равен 10. Микросхема 74OL6000 (RS650-829) – это буфер ТТЛШ–ТТЛ, 74OL6001 (RS650-835) – преобразователь ТТЛШ–ТТЛ, 74OL6010 (RS650-841) – буфер ТТЛШ–КМОП и 74OL6011 (RS650-857) – преобразователь ТТЛШ–КМОП.

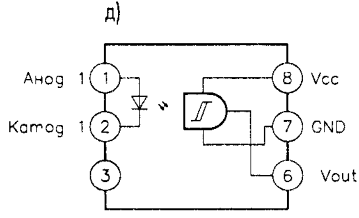


Рис. 4.26. Различные типы оптопар а – HCPL-2630, б – внутренняя блок-схема HCPL-2630, в – 74OL6000 и 74OL6001, г – 74OL6010 и 74OL6011, д – H11L1 и H11N1

Некоторые оптопары, в частности H11L1 (Isocom Components, RS585-292) и H11N1 (Quality Technologies, RS577-875), имеют *выход на триггере Шмитта* (рис. 4.26д). Скорость передачи данных 1 Мб/с (H11L1) или 5 Мб/с (H11N1). Прямое напряжение на светодиоде 1,5 В, максимальный прямой ток 60 мА для H11L1 и 30 мА для H11N1. Максимальный ток коллектора 50 мА.

5. УПРАВЛЕНИЕ ВНЕШНИМИ УСТРОЙСТВАМИ

В приложениях, о которых идет речь в данной книге, компьютер должен управлять множеством внешних устройств. Это могут быть осветительные приборы, нагреватели, электрические двигатели переменного тока, звуковые колонки, мониторы и т.д.

5.1. Мощные устройства коммутации

Цифровые интегральные микросхемы, как правило, не способны генерировать большой ток для управления внешними приборами. В таких системах требуются мощные исполнительные устройства.

5.1.1. Устройства коммутации на оптопарах

Оптопары можно использовать для управления приборами с малыми значениями токов, требующими гальванической развязки. Максимальный рабочий ток ограничен характеристиками фототранзисторов. Например, оптопары Дарлингтона серии PC815 (Sharp, RS175-198) имеют максимальное значение выходного тока 80 мА, достаточное для управления слаботочным реле, которое, в свою очередь, способно работать с более мощными приборами (рис. 5.1). Оптопары Дарлингтона серии PS2502 (NEC, RS590-424 и RS590-430) поддерживают токи до 160 мА.

Коэффициент передачи обычно достигает 2000%. Управляющее светодиодом напряжение равно 1,1 В, максимальный рабочий ток – 80 мА. Максимальное напряжение между коллектором и эмиттером фототранзистора составляет 40 В, а время насыщения – 100 мкс.

5.1.2. Транзисторные устройства коммутации

В данном разделе речь пойдет о простейшем и наиболее экономичном способе управления внешними приборами. В частности, применяются n–p–n транзисторы

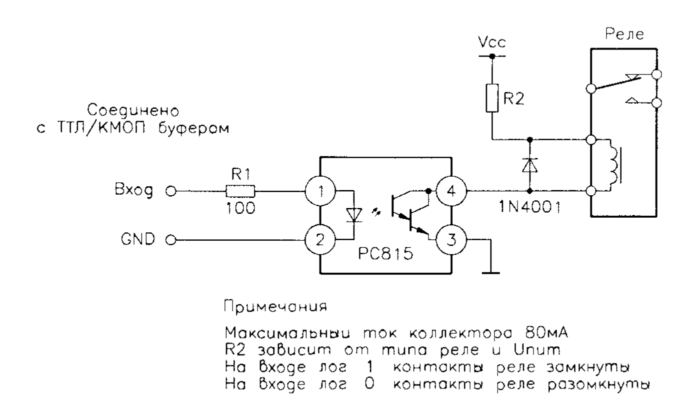


Рис. 5.1. Оптопара PC815 для управления реле

BC108C и ZTX300 (рис. 5 2а), которые имеют максимальные значения тока коллектора 100 и 500 мА с максимальным уровнем мощности в 300 и 500 мВт. Максимальное напряжение между коллектором и эмиттером равно 20 В для BC108C и 25 В для ZTX300. Максимальная рабочая частота – соответственно 300 и 150 МГц. При работе на индуктивную нагрузку, такую как реле или электродвигатель, следует применять защитные диоды; при работе на активную нагрузку их использование необязательно.

5.1.3. Устройства коммутации на основе схемы Дарлингтона

На рис. 5.2б изображена схема на базе мощного транзистора Дарлингтона TIP122 или TIP142 TIP122 управляет напряжениями до 100 В и токами до 5 А. Максимальная рассеиваемая мощность – 65 Вт. Транзисторы Дарлингтона открываются при напряжении 1,2 В между базой и коллектором и могут усиливать ток в 5000 раз. Следовательно, напряжение на базе, превышающее 1,2 В, вызовет режим насыщения транзистора База соединяется с ТТЛ входом через резистор. TIP142 функционирует при коллекторном токе 10 А. Максимальная рабочая частота этих транзисторов равна 5 МГц. Для индуктивной нагрузки должны использоваться защитные диоды.

5.1.4. Устройства коммутации на полевых транзисторах

На рис. 5.2в изображена схема на базе полевых МОП транзисторов VN10KM или VN66AF. Чтобы МОП транзистор открылся, на него необходимо подать прямое напряжение смещения порядка 0,8 В. Прямое смещение порядка 5 В существенно влияет на проводимость. Поскольку входное сопротивление устройства на полевом транзисторе крайне высоко, соединить его напрямую с выходным портом компьютера невозможно Таким устройством следует управлять через резистор. VN10KM выдерживает максимальное напряжение 60 В и ток 310 мА, VN66AF работает при максимальном напряжении 60 В и токе 2 А. Время насыщения – около 15 нс.

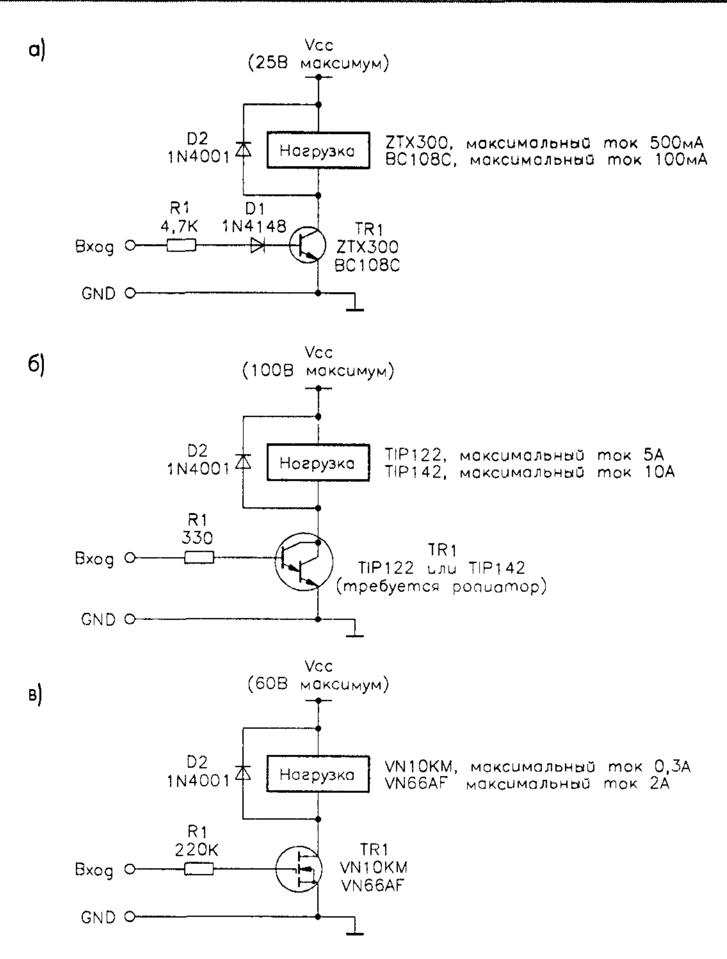


Рис. 5.2. Устройства управления: а – на базе биполярного транзистора; б – на транзисторе Дарлингтона; в – на полевом транзисторе

5.1.5. Устройства коммутации на МОП транзисторах с защитой

Мощные устройства управления на базе МОП транзисторов с защитой, иногда называемые *твердотельными реле*, играют роль переключателей в силовых цепях цифровых систем управления. Входное управление совместимо с пятивольтовыми логическими уровнями. В этих элементах используется встроенная схема термоконтроля, которая защищает их от перегрева, короткого замыкания и больших величин тока. Она перекрывает выход при температуре 140 °С, а когда температура падает до 125 °С, термозащита выключается. Такие устройства, как правило, имеют информационный выход, который низким уровнем сигнализирует о срабатывании встроенных цепей защиты.

Транзистор BTS410 (рис. 5.3) способен управлять напряжениями в диапазоне 4,9–40 В, порог срабатывания защиты от перенапряжения порядка 42–52 В. Максимальная рабочая температура равна 150 °С. В зависимости от температуры уровень превышающих значений тока колеблется в пределах 3,1–21 А. Это устройство имеет низкое сопротивление во включенном состоянии во всем диапазоне температур. Время включения и выключения составляет 60 и 50 мкс соответственно. Входное напряжение включения изменяется от 2 до 5 В, выключения – от 0 до 0,8 В. Входной ток равен 25 мкА при входном напряжении 3,5 В.

Два других примера таких устройств – VN05N и VN20N, их выходы и типовая схема включения показаны на рис. 5.3. VN05N рассчитан на выходной ток 12 А, а VN20N – на 28 А.

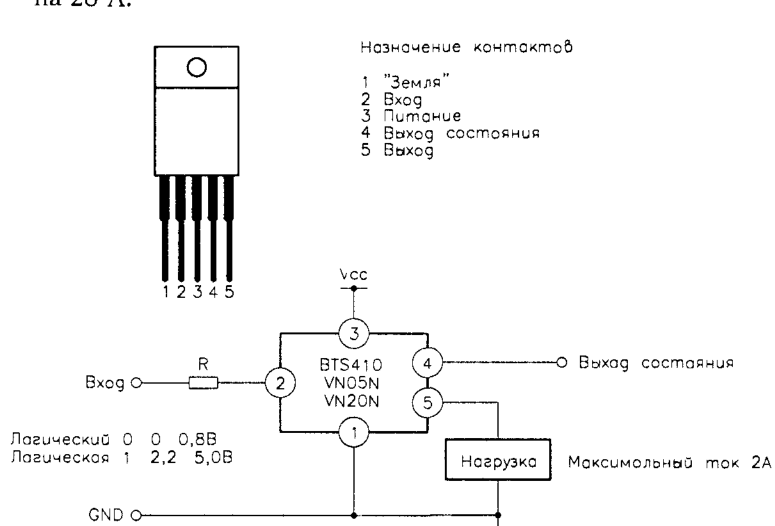


Рис. 5.3. Устройство управления на МОП транзисторе с защитой

5.2. Устройства управления светодиодами

Светодиод – один из самых распространенных приборов, он применяется для сигнализации и индикации различных состояний электронных устройств. В данном разделе приводятся принципиальные электрические схемы, которые позволяют управлять светодиодами.

5.2.1. Стандартные светодиоды

Стандартные светодиоды потребляют ток от 10 до 20 мА при напряжении 2 В. На рис. 5.4а изображено устройство управления на базе транзистора ZTX300. В схеме последовательно со светодиодом используется резистор; его величина выбирается в зависимости от подаваемого напряжения. Светодиоды также могут управляться через ТТЛ или КМОП входы напрямую (рис. 5.4б,в). Подаваемое напряжение для интегральных схем ТТЛ составляет +5 В. Номинал последовательного резистора

должен быть порядка 4,7 кОм. Для КМОП схем сопротивление следует устанавливать в соответствии с подаваемым напряжением (рис. 5.4).

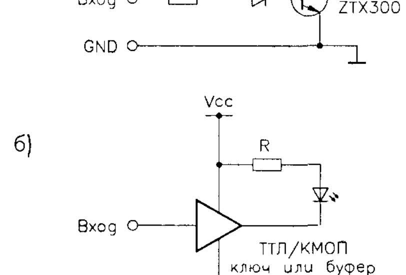
5.2.2. Маломощные светодиоды

Маломощные светодиоды потребляют ток 2 мА при напряжении 1,8 В. Управление ими осуществляется с помощью схем, предназначенных для стандартных светодиодов (см. рис. 5.4). Однако величины последовательных сопротивлений должны быть другими. Соотношения между напряжением питания (Vcc) и величиной сопротивления (R) приведены ниже:

Vcc = 3–4 В R = 600 Ом
Vcc = 4–5 В R = 1,6 кОм
Vcc = 5–8 В R = 3,1 кОм
Vcc = 8–12 В R = 5,1 кОм
Vcc = 12–15 В R = 6,6 кОм

5.2.3. Многоцветные светодиоды

Существует два типа многоцветных светодиодов. Первый – это двухцветные, в корпусе которых совмещены зеленый и красный светодиоды. Изменяя полярность,



Vcc (В)	R (Ом)
3 4	100
4 5	150
5 8	220
8 12	470
12 15	820

Vcc (В)	R (Ом)
3 4	0
4 5	100
5 8	220
8 12	470
12 15	820

Vcc (В)	R (Ом)
3 4	0
4 5	100
5 8	220
8 12	470
12 15	820

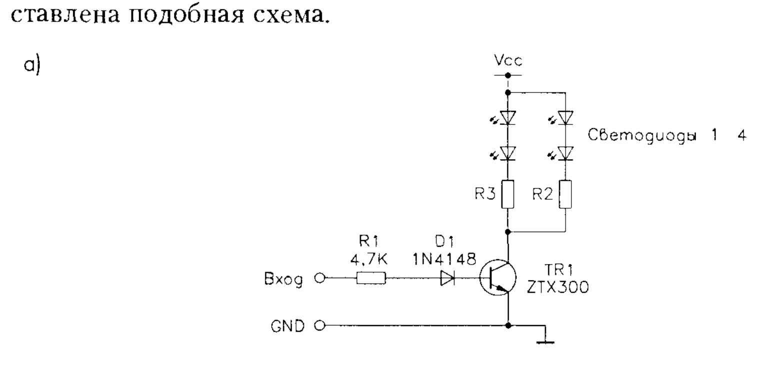
Рис. 5.4. Схемы управления светодиодами а – светодиод горит, когда на входе 1, б – светодиод горит, когда на входе 0, в – светодиод горит, когда на входе 1

можно менять цвет с красного на зеленый и наоборот. Второй тип – трехцветные с тремя контактами: один общий, а два других подсоединены к анодам красного и зеленого светодиодов.

Трехцветные светодиоды имеют четыре состояния: оба выключены, красный включен, зеленый включен или оба включены (дают желтый цвет). Их управление обеспечивают схемы, изображенные на рис. 5.4.

5.2.4. Инфракрасные светодиоды

Большинство инфракрасных светодиодов работает со значительными величинами токов и чаще всего применяется для дистанционного управления и связи. Пример таких светодиодов – серия SFH485 (Siemens). Максимальный прямой ток через диод составляет 100 мА при напряжении 1,5 В. SFH485 (RS585-242) имеет ширину диаграммы направленности луча 40°, а SFH485P (RS585-236) – 80°. Мощность излучения для SFH485 равна 16–32 мВт, а для SFH485P – 3,15–6,3 мВт. Посредством схемы (см. рис. 5.4) можно управлять инфракрасными светодиодами. Для увеличения мощности ИК следует использовать несколько светодиодов, соединенных параллельно или последовательно. На рис. 5.5а представлена подобная схема.



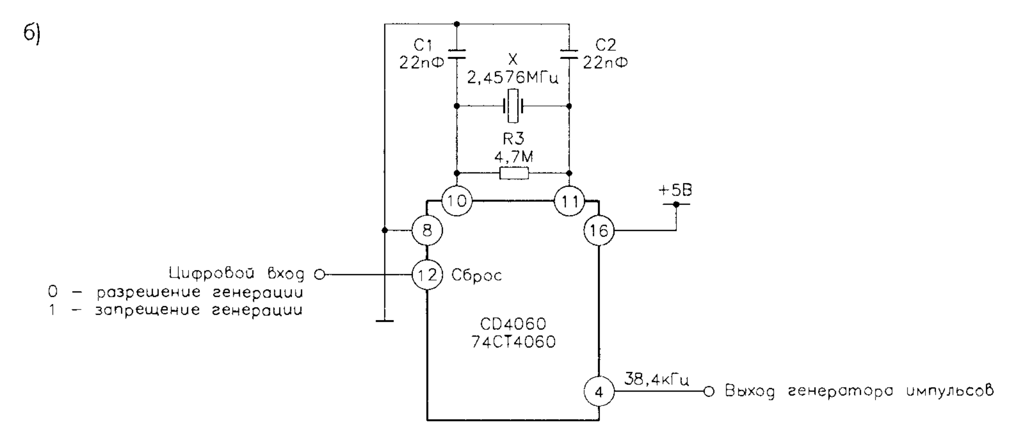


Рис. 5.5. Устройства управления инфракрасными светодиодами в приложениях дистанционного управления: а – устройство управления для нескольких светодиодов; б – импульсный генератор

Серия OD880 (Optek) дает большую мощность светового излучения. Ширина диаграммы направленности луча равна 80° для OD880W (RS195-439), 35° для OD880L (RS195-445) и 8° для OD880F (RS195-451). Максимальное прямое напряжение составляет 1,9 В при токе 100 мА. При работе с импульсным сигналом пиковое значение тока достигает 3 А. Мощность излучения этих устройств соответственно 16, 50 и 135 мВт.

Импульсная схема инфракрасного излучения часто используется в приложениях дистанционного управления. Электрический ток, проходящий через светодиод, представляет собой последовательность импульсов вместо постоянного тока. Когда светодиод находится в открытом состоянии, через него проходит большой ток. При таком режиме работы образуется повышенная излучаемая мощность. На рис. 5.5б изображен генератор импульсов с частотой 38,4 кГц. Для управления светодиодами могут применяться транзисторы соответствующих типов.

5.3. Устройства управления реле

Реле – распространенный прибор для управления мощными исполнительными устройствами. В данном разделе приводятся принципиальные электрические схемы, совместно с которыми реле может использоваться.

5.3.1. Реле с сухими контактами

Маломощные реле с сухими контактами работают с напряжениями порядка 3,7 В и токами 7,4 мА, их управление осуществляется непосредственно микросхемами

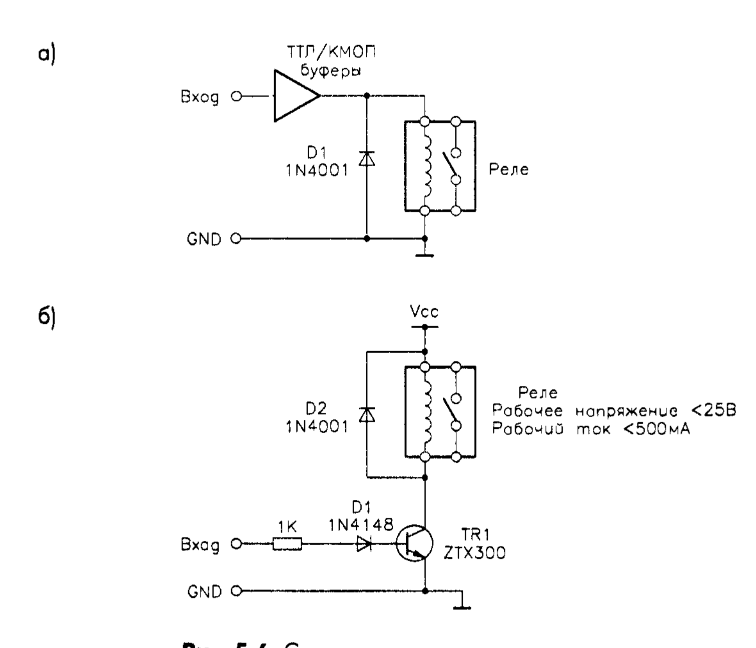


Рис. 5.6. Схемы управления реле с сухими контактами: а – на ТТЛ ключах; б – на транзисторе

ТТЛ (рис. 5.6а). Когда на вход ТТЛ подается высокий уровень, контакты реле замыкаются. Для защиты микросхемы должны использоваться диоды. Максимальное коммутируемое напряжение обычно не превышает 240 В.

5.3.2. Транзисторные устройства управления реле

Реле средней и большой мощности требуют больших напряжений и токов через обмотку. На рис. 5.6б показана схема на базе транзистора ZTX300. Напряжение питания схемы равно 25 В, потребляемый ток 0,5 А. Напряжение питания зависит от типа используемого реле.

5.4. Мощные управляющие интегральные микросхемы

В этом разделе дается описание многоканальных электронных приборов, которые, как и реле, предназначены для управления различными исполнительными устройствами, но характеризуются большим быстродействием.

5.4.1. Многоканальные управляющие интегральные микросхемы

Когда в приложениях управления требуются многоканальные устройства, можно использовать управляющие интегральные микросхемы типа матрицы ULN2803A (SGS-Thomson), имеющей восемь ключей на транзисторах Дарлингтона (рис. 5.7). Каждый ключ способен коммутировать ток до 500 мА при напряжении до 50 В. Входы микросхемы соединяются напрямую с ТТЛ/КМОП схемами. ULN2003A (Allegro Microsystems, RS307-109) имеет семь ключей.

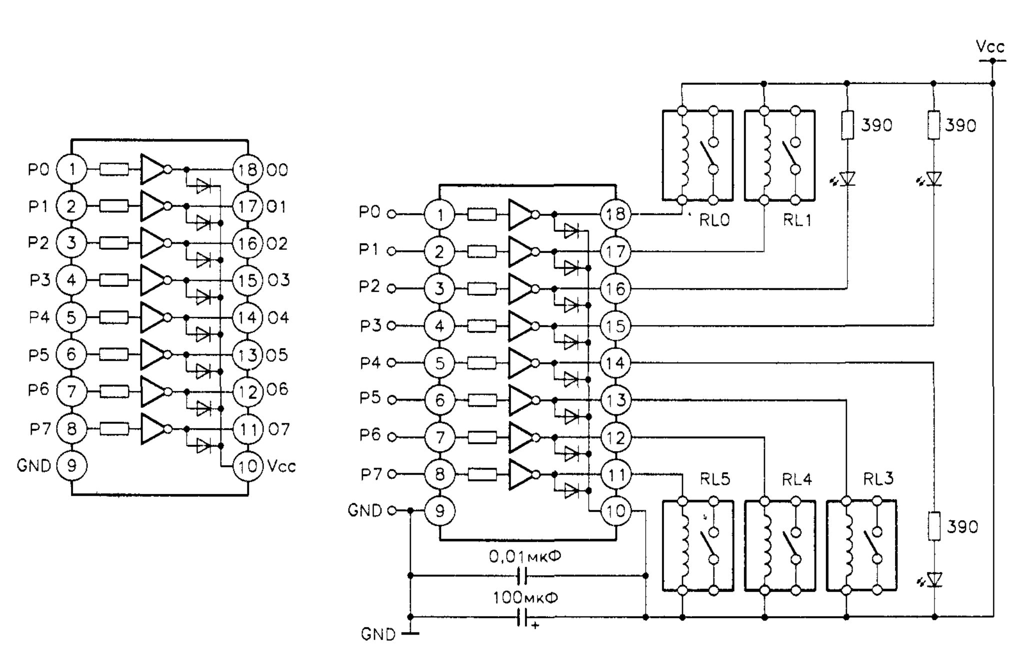


Рис. 5.7. Транзисторная матрица Дарлингтона ULN2803

5.4.2. Буферные устройства управления с защелками

UCN5832A (Allegro Microsystems, RS426-755) представляет собой 32-канальное устройство управления с защелкой и последовательным входом (рис. 5.8). Оно имеет 32 биполярных n–p–n транзистора с открытым коллектором. Каждый транзистор способен коммутировать ток до 150 мА при напряжении до 40 В. Микросхема содержит 32-канальный буфер-защелку, два высокоскоростных 16-разрядных

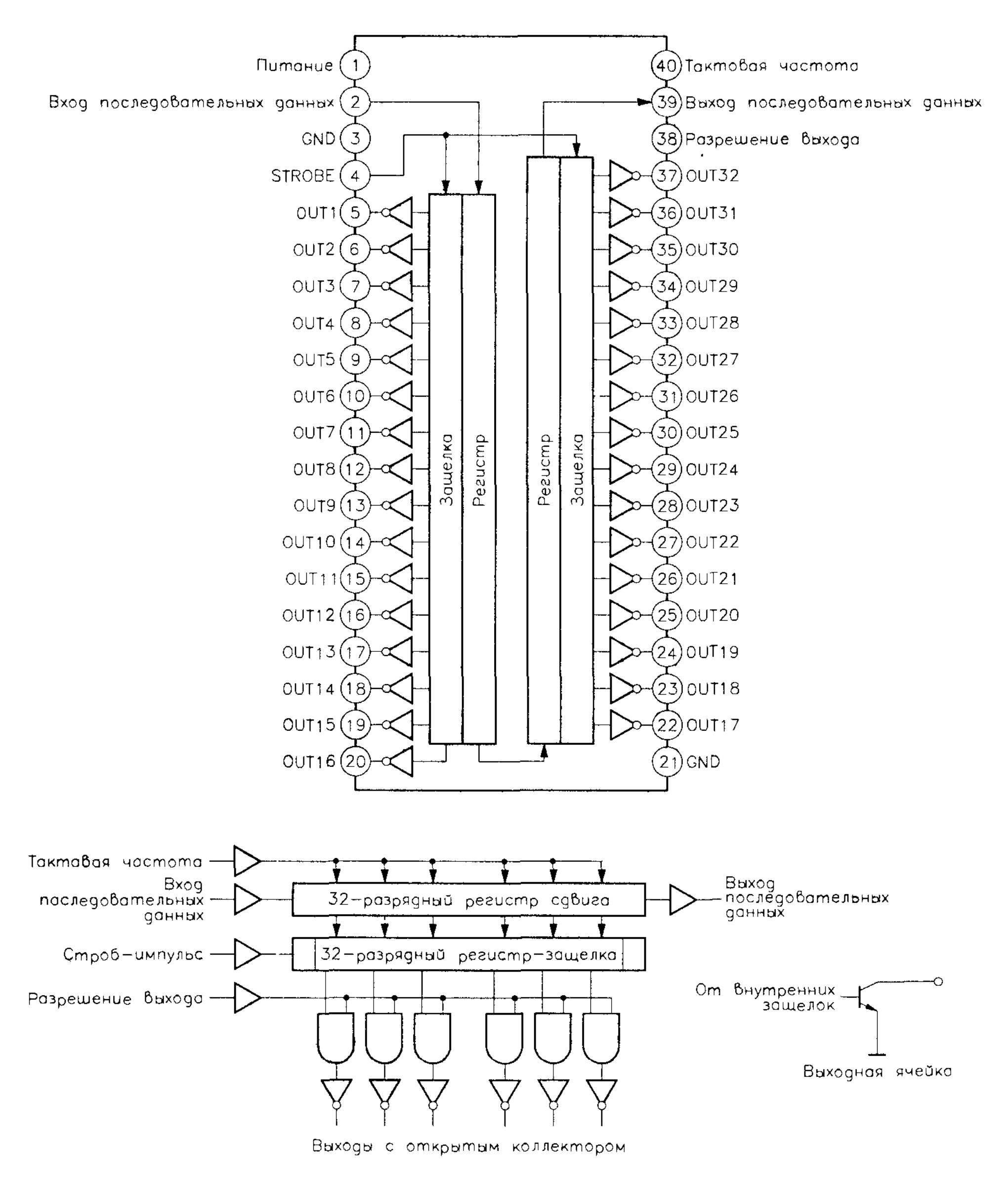


Рис. 5.8. Назначение выводов и внутренняя блок-схема UCN5832A

регистра сдвига и схемы управления. Управление осуществляется через четыре КМОП входа, которые могут контролироваться непосредственно с выходов компьютера. Если управляющие входы соединены с выходами ТТЛ, необходимо использовать нагрузочные резисторы 4,7 кОм. Максимальная скорость входных данных – 3,3 Мб/с.

Временные диаграммы изображены на рис. 5.9. Входные данные загружаются в регистр сдвига по положительному фронту тактового импульса. При прохождении следующих тактовых импульсов данные сдвигаются по направлению к выходу последовательных данных (контакт 39). Входные данные остаются неизменными непосредственно перед фронтом тактового импульса. Информация из регистра передается в соответствующий буфер-защелку при подаче на стробирующий вход (контакт 4) сигнала высокого уровня. Буфер удерживает данные до тех пор, пока сигнал на этом контакте не изменится. Данные появляются на выходах микросхемы во время отрицательного фронта стробирующего импульса. Когда на контакте 38 (разрешение выхода) сигнал низкого уровня, все выходы буфера отключены. Если на него подать сигнал высокого уровня, то на выходы будет выгружено содержимое буфера.

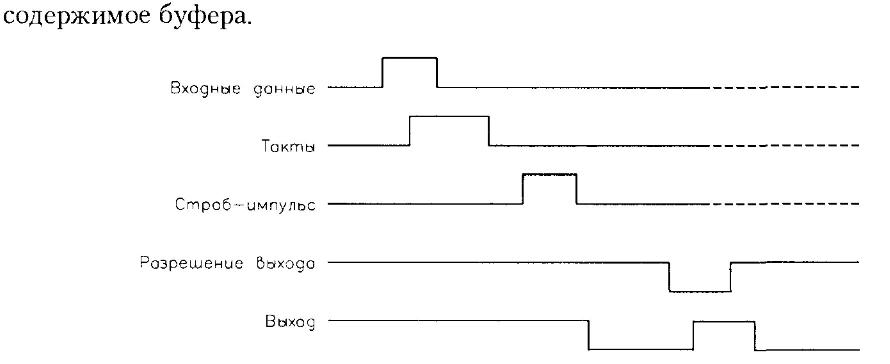


Рис. 5.9. Временные диаграммы работы микросхемы UCN5832A

Схема с использованием параллельного порта изображена на рис. 5.10. Последовательный, тактирующий и стробирующий входы соединены соответственно с контактами D1, D2 и D3 на экспериментальной плате параллельного порта. Текст программы управления на языке TP6 приведен ниже.

Текст программы 5832.PAS

```
Program 5832A;
(*Программа управления микросхемой UCN5832A.*)
(* Соединена с экспериментальной платой параллельного порта,
   последовательный вход данных соединен с D1,
   тактирующий вход соединен с D2,
   стробирующий вход соединен с D3.*)
uses
   crt,dos,
{$I c.\ioexp\tplib1.pas}

var
   bank1,bank2,bank3,bank4:byte;
```

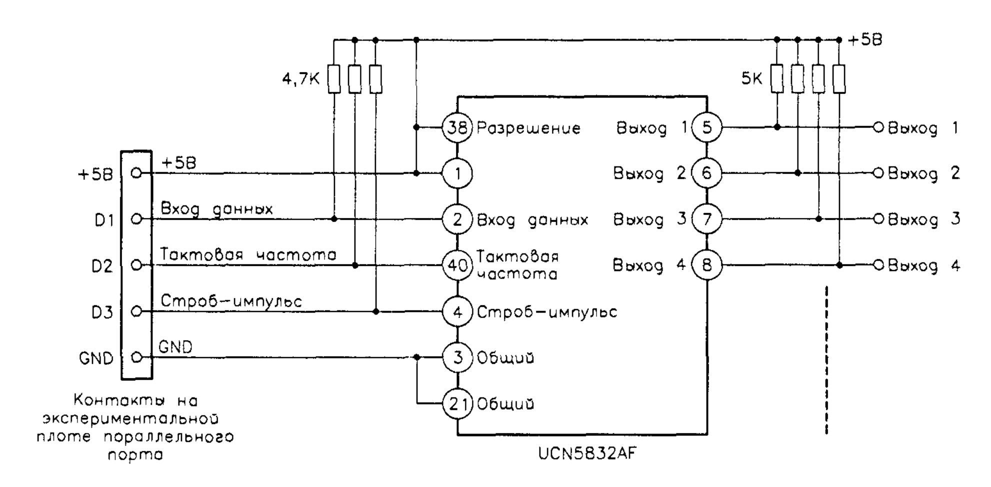


Рис. 5.10. Экспериментальная схема на базе UCN5832AF

```
procedure load_data(x byte),
(*Загрузка данных в UCN5832A *)
begin
  write_data_port(p_address,x),        (*Данные=x, такты=0, строб-импульс=0 *)
  write_data_port(p_address,x+2),      (*Данные=x, такты=1, строб-импульс=0.*)
  write_data_port(p_address,x),        (*Данные=x, такты=0, строб-импульс=0.*)
end,

procedure strobe,
(*Стробирование.*)
  write_data_port(p_address,4),        (*Строб=1 *)
  write_data_port(p_address,0),
end,

procedure output_control(bank1,bank2,bank3,bank4 byte);
(*bank1 - младший байт, bank4 - старший байт.*)
var
  i:integer;
begin
  for i:=1 to 8 do load_data(round(bank4 and bit_weight(9-i)/bit_weight(9-i)));
  for i:=1 to 8 do load_data(round(bank3 and bit_weight(9-i)/bit_weight(9-i)));
  for i =1 to 8 do load_data(round(bank2 and bit_weight(9-i)/bit_weight(9-i)));
  for i.=1 to 8 do load_data(round(bank1 and bit_weight(9-i)/bit_weight(9-i)));
strobe,      (*Сигнал стробирования для записи данных в UCN5832A *)
end,

(*Главная программа *)
begin
  centronic_address;
  writeln('UCN5832A demonstration program );
  writeln('Out1,Out9,Out17 and Out25 oscillating, other outputs=zero');
  repeat
     output_control(1,1,1,1), (*Сначала каждый байт равен 1  Включен режим вывода данных
                               (выход инвертирован).*)
```

```
      delay(1000),
      output_control(0,0,0,0), (*Закрытие режима вывода данных *)
      delay(1000),
   until keypressed,
end
```

Микросхема UCN5810AF (Allegro Microsystems) – это 10-канальный буфер-защелка (рис. 5.11). Каждый канал может коммутировать ток 15 мА. Ток потребления в статическом режиме при напряжении питания 5 В равен 100 мкА. Логика работы микросхемы такая же, как и UCN5832A, однако на выходах сигнал не инвертируется.

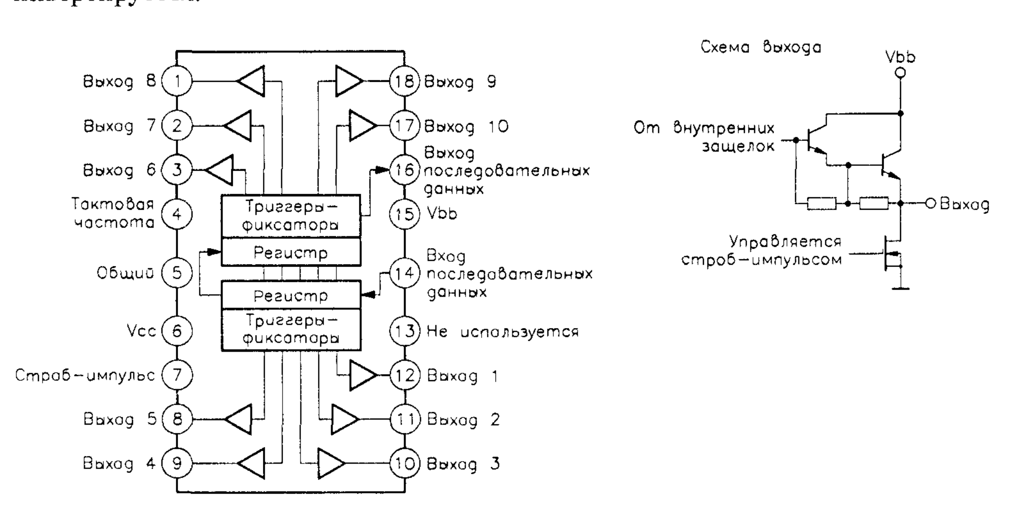


Рис. 5.11. Назначение выводов и внутренняя блок-схема UCN5810AF

5.5. Оптоэлектронные полупроводниковые реле на тиристорах

Полупроводниковые реле применяются в качестве коммутаторов в силовых цепях переменного тока. При подаче на вход сигнала высокого уровня загорается светодиод. Его световой поток управляет фототранзистором, который, в свою очередь, управляет тиристором, используемым в качестве коммутационного прибора. Когда тиристор включен, устройство проводит ток. Поскольку связь между входом и выходом устанавливается с помощью светового потока, разница потенциалов между ними может достигать нескольких тысяч вольт. Для таких устройств характерно наличие детектора нулевого напряжения, который включает тиристор только в том случае, когда переменное напряжение близко к нулю.

Реле MOC3041 (ISOCOM Components, RS195-4122) содержит инфракрасный светодиод, детектор нулевого напряжения и тиристор. Блок-схема и схема включения приведены на рис. 5.12. Максимально коммутируемое напряжение составляет 400 В при токе 100 мА. На рис. 5.13 изображено мощное полупроводниковое реле с детектором нулевого напряжения.

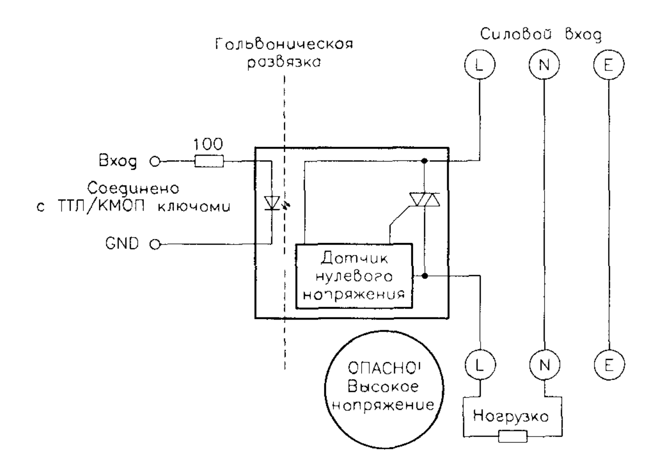


Рис. 5.12. Оптоэлектронное реле для приложений управления в силовых цепях переменного тока

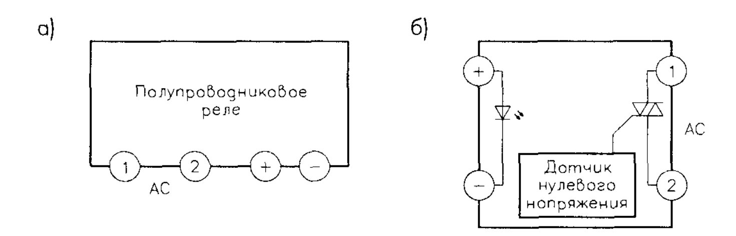


Рис. 5.13. Мощное полупроводниковое реле с детектором нулевого напряжения а – контакты, б – внутренняя блок-схема

5.6. Устройства управления двигателями постоянного тока

Двигателями постоянного тока можно управлять с помощью реле (см. рис. 5.6) или транзисторов (см. рис. 5.2). Одиночное переключающееся реле включает и выключает двигатель, а спаренное отвечает за направление вращения (рис. 5.14).

Другой способ управления двигателями постоянного тока основан на использовании *мостовых схем* типа L298N (SGS-Thomson, RS636-384). Это мощное – напряжение до 46 В, ток до 2 А на каждый канал – двухканальное устройство, которое работает от уровней ТТЛ (рис. 5.15).

С вывода Vs (контакт 4) поступает напряжение питания для двигателя, на вывод Vss (контакт 9) подается напряжение питания схемы (+5 В). Выводы ENA и ENB (контакты 6 и 11) открывают входы двух каналов. Входы IN1 и IN2 (контакты 5 и 7) управляют первым каналом, а IN3 и IN4 – вторым. Эмиттеры транзисторов соединены для подключения внешних контролирующих датчиков. Типовая схема включения для одного канала показана на рис. 5.16.

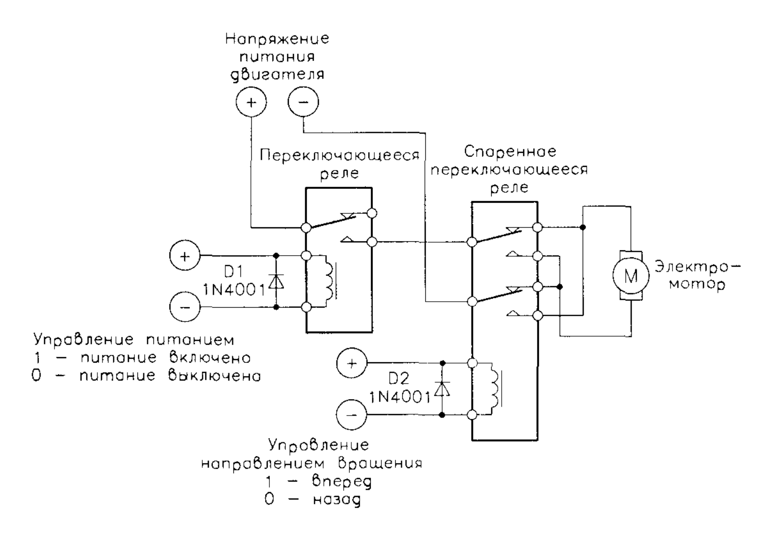
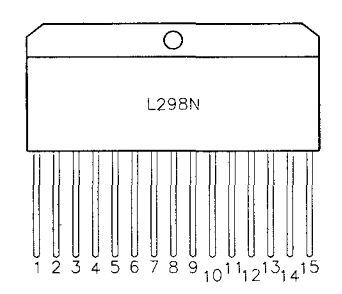


Рис. 5.14. Устройства управления двигателем постоянного тока посредством реле



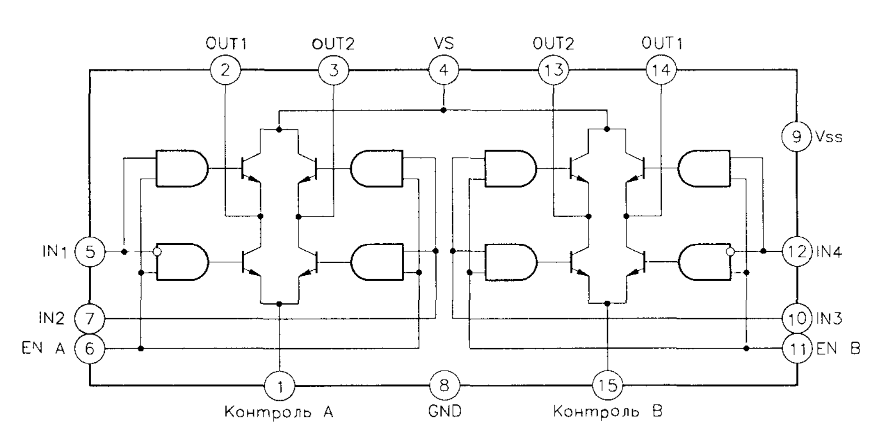


Рис. 5.15. Расположение выводов и внутренняя блок-схема мостового устройства управления L298N

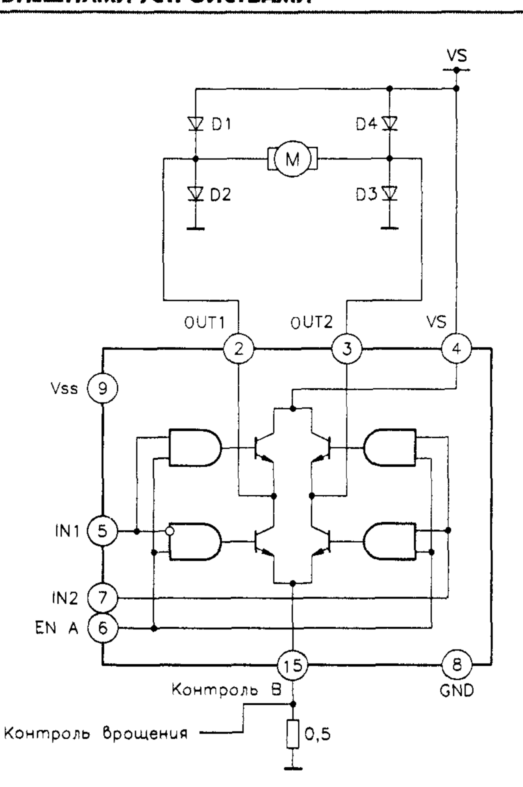


Рис. 5.16. Устройство управления двигателем постоянного тока

Когда на входе ENA низкий уровень, входы заблокированы и двигатель не вращается. Если на этот вход подать высокий уровень, входы открываются. Входы IN1 и IN2 управляют режимами работы двигателя следующим образом:

- IN1 = 1, IN2 = 0 – двигатель вращается по часовой стрелке;
- IN1 = 0, IN2 = 1 – двигатель вращается против часовой стрелки;
- IN1 = IN2 – двигатель не вращается.

5.7. Устройства управления шаговыми двигателями

Существует два типа шаговых двигателей: *четырехфазные* и *двухфазные* (рис. 5.17). Для этих двигателей требуются различные схемы управления.

5.7.1. Устройства управления четырехфазными шаговыми двигателями

Для управления шаговыми двигателями используются различные алгоритмы, которые отличаются друг от друга последовательностями возбуждения обмоток. Всего существует три *шаговые последовательности*: волновая, полушаговая и шаговая. Порядок подачи напряжения на обмотки двигателя приведен на рис. 5.18.

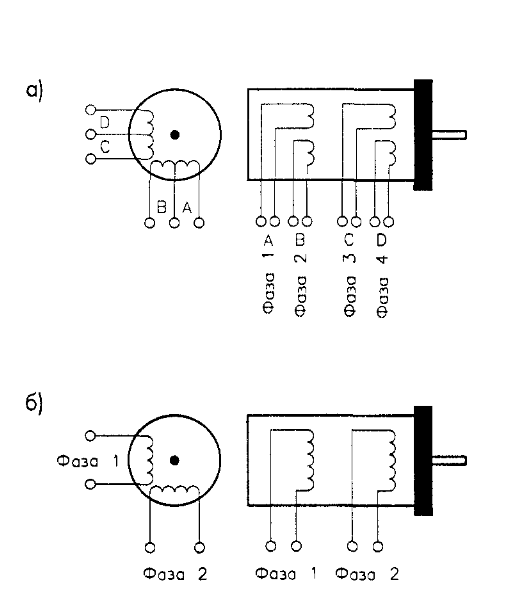


Рис. 5.17. Шаговые двигатели. а – четырехфазный, б – двухфазный

а)

Шаг	A	B	C	D
1	ON	OFF	OFF	OFF
2	OFF	ON	OFF	OFF
3	OFF	OFF	ON	OFF
4	OFF	OFF	OFF	ON

б)

Шаг	A	B	C	D
1	ON	OFF	OFF	ON
2	ON	ON	OFF	OFF
3	OFF	ON	ON	OFF
4	OFF	OFF	ON	ON

в)

Шаг	A	B	C	D
1	ON	OFF	OFF	OFF
2	ON	ON	OFF	OFF
3	OFF	ON	OFF	OFF
4	OFF	ON	ON	OFF
5	OFF	OFF	ON	OFF
6	OFF	OFF	ON	ON
7	OFF	OFF	OFF	ON
8	ON	OFF	OFF	ON

Рис. 5.18. Шаговые последовательности запуска четырехфазного шагового двигателя: а – волновая, б – полушаговая; в – шаговая

Волновая последовательность возбуждения – самый простой способ управления шаговыми двигателями: обмотки возбуждаются одна за другой. При этом двигатель начинает вращаться в сторону, противоположную порядку возбуждения обмоток. Так как в одно время возбуждается только одна обмотка, вращающий момент двигателя небольшой. Для его увеличения используется *шаговая последовательность*, которая аналогична предыдущей, но здесь одновременно возбуждаются две обмотки, благодаря чему увеличивается вращающий момент двигателя.

Полушаговая последовательность возбуждения – это комбинация первых двух. Во время одного оборота ротора количество циклов возбуждения удваивается. При этом режиме двигатель работает более ровно. Для управления двигателями существуют специальные микросхемы.

Микросхема UCN5804 (Allegro Microsystems, RS653-531) может генерировать все три последовательности (рис. 5.19). Для ее работы требуются два источника питания: один для самой микросхемы (контакт 16), максимальное напряжение 7 В, другой для управления двигателем (контакты 2 и 7). Контакты 4, 5, 12 и 13 – «земля» источников питания. Контакты 1, 3, 6 и 8 конструктивно соединены с транзисторами Дарлингтона (максимальное напряжение 35 В, максимальный ток 1,5 А).

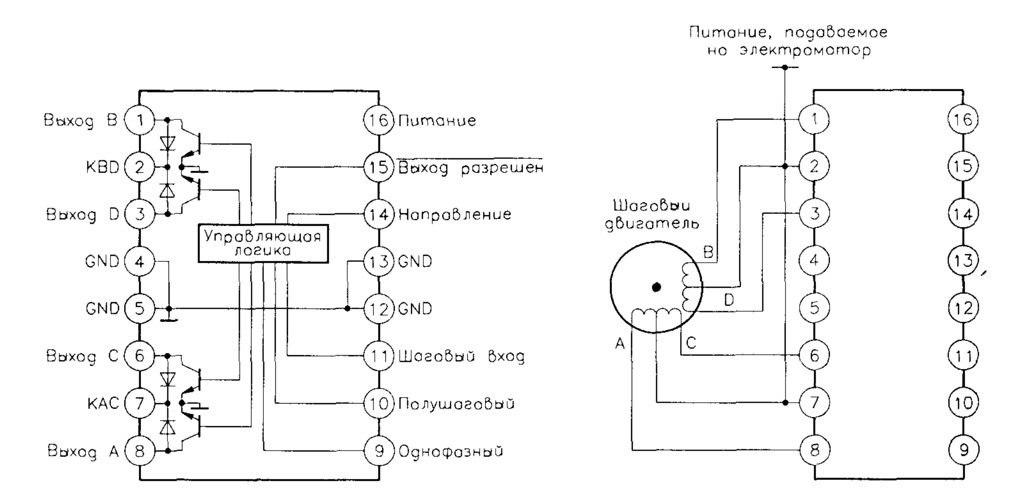


Рис. 5.19. Назначение выводов и типовое включение микросхемы UCN5804

Контакт 15 (Выход разрешен) управляет микросхемой. Когда на него подан сигнал высокого уровня, все выходы закрыты. Контакт 14 (направление) устанавливает направление вращения двигателя; контакт 11 – это «шаговый вход»: отрицательный фронт сигнала, поданный на него, поворачивает двигатель на один шаг. Режимы возбуждения двигателя устанавливаются с помощью контактов 9 и 10:

- контакт 9 = 0, контакт 10 = 0 – шаговый;
- контакт 9 = 1, контакт 10 = 0 – волновой;
- контакт 9 = 0, контакт 10 = 1 – полушаговый;
- контакт 9 = 1, контакт 10 = 1 – блокировка.

Во время работы состояние выводов 9, 10 и 14 можно изменить, если на «шаговом входе» логическая единица.

Хорошо известна и микросхема SAA1027 (Philips Semiconductor, RS300-237), которая применяется для управления четырехфазными двигателями. Напряжение питания должно быть от 9,5 до 18 В. Максимальный выходной ток – 500 мА. Микросхема несовместима с ТТЛ логикой. Напряжение выше 7,5 В определяет уровень логической единицы, а напряжение ниже 4,5 В – логического нуля.

Для управления мощными двигателями можно использовать транзисторы Дарлингтона типа TIP122. Номинальный ток такого транзистора равен 5 А при напряжении до 100 В. Схема устройства управления изображена на рис. 5.20. Входы A, B, C и D соединяются с компьютером при помощи описанных выше микросхем управления UCN5804 или SAA1027 через схему сопряжения.

5.7.2. Устройства управления двухфазными шаговыми двигателями

Схема управления двухфазным шаговым двигателем изображена на рис. 5.21. Мостовая микросхема L298N используется для управления питанием двигателя, контроллер L297 (SGS-Thomson, RS636-362) – для генерации шаговых последовательностей.

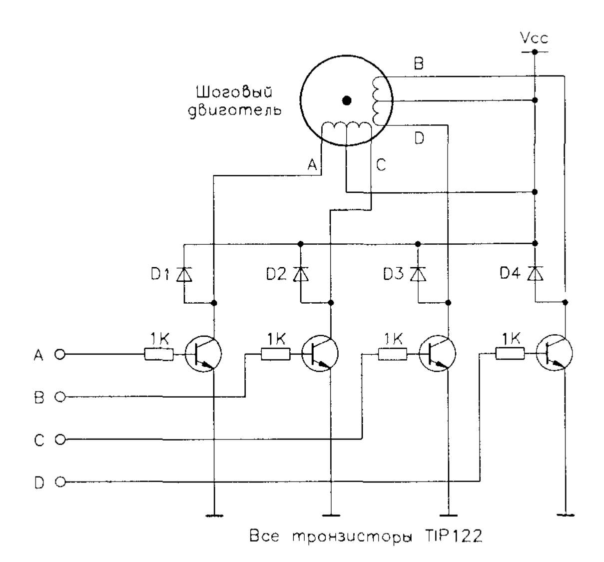


Рис. 5.20. Устройство управления шаговыми двигателями с использованием транзисторов Дарлингтона

Рис. 5.21. Схема управления двухфазным шаговым двигателем

5.8. Управление звуковыми устройствами

В разделе дается описание принципиальных электрических схем и электронных приборов для управления громкоговорителями, сиренами и другими устройствами, предназначенными для генерации и усиления звуковых колебаний.

5.8.1. Устройства управления пьезоэлектрическими динамиками, зуммерами и сиренами

Пьезоэлектрические динамики служат для генерации звуков. Они имеют максимальное входное напряжение 50 В и номинальный ток 10 мА. На рис. 5.22а изображена схема, использующая буфер КМОП/ТТЛ для управления таким динамиком. Схема транзисторного устройства управления ZTX300 показана на рис. 5.22б.

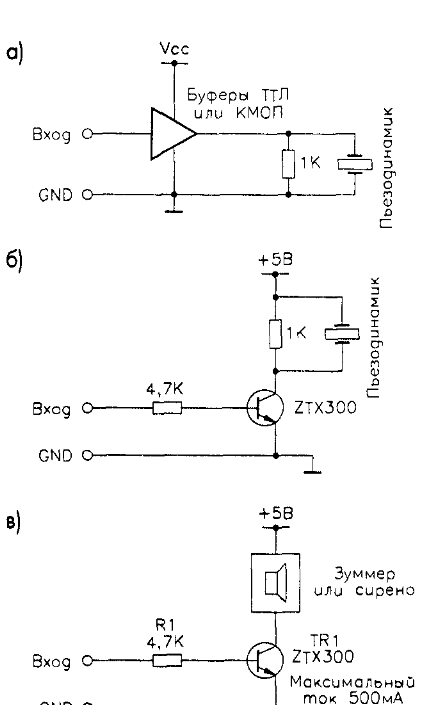


Рис. 5.22. Схемы управления для звуковых устройств а – на базе ТТЛ/КМОП; б – на транзисторе; в – управление зуммером или сиреной

Чтобы получить звук, необходимо подать на вход последовательность импульсов. *Полупроводниковые зуммеры* – это автономные динамики, способные генерировать тон частотой порядка 450 Гц. На рис. 5.22в представлена схема управления на транзисторе ZTX300. Для генерации звука на базу ZTX300 необходимо подать высокий уровень напряжения. При управлении сиренами можно использовать такие же схемы.

Ультразвуковые преобразователи предназначены для генерации ультразвука. Обычно они применяются в приложениях дистанционного управления, измерения и передачи данных, например в ультразвуковом измерителе расстояний и детекторах движения объекта. На рис. 5.23 изображена схема, генерирующая сигнал частотой 38,4 кГц.

В данной схеме ультразвуковой сигнал генерируется только в том случае, когда на контакт 12 (RESET) подается сигнал низкого уровня. Если на этом входе высокий уровень, генерация подавляется.

5.8.2. Устройства управления громкоговорителями

На рис. 5.24 показана схема управления громкоговорителем на базе микросхемы усилителя низкой частоты TBA820M. Напряжение питания микросхемы может изменяться от 3 до 12 В, выходная мощность составляет 2 Вт для громкоговорителя с сопротивлением 8 Ом. Контакт 2 – инвертирующий вход. Между входом и выходом усилителя подключен внутренний резистор 6 кОм, играющий роль отрицательной обратной связи. Коэффициент усиления по напряжению определяется как отношение сопротивлений внутреннего и внешнего резисторов. Его рекомендуемая величина от 22 до 220 Ом (резистор R2).

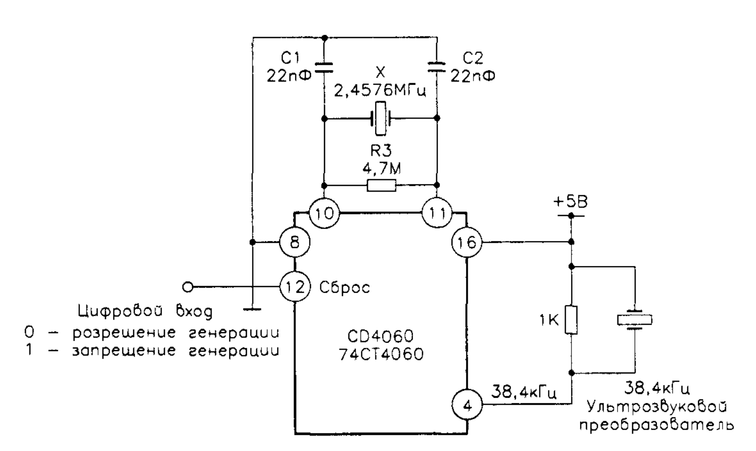
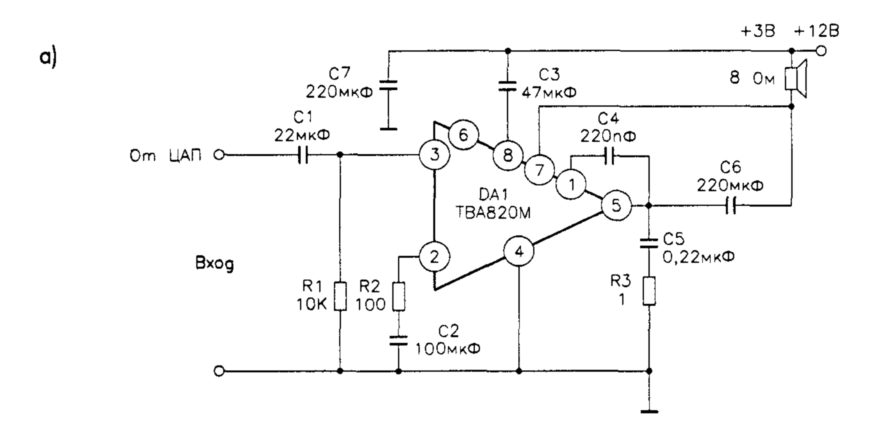


Рис. 5.23. Схема генератора ультразвука



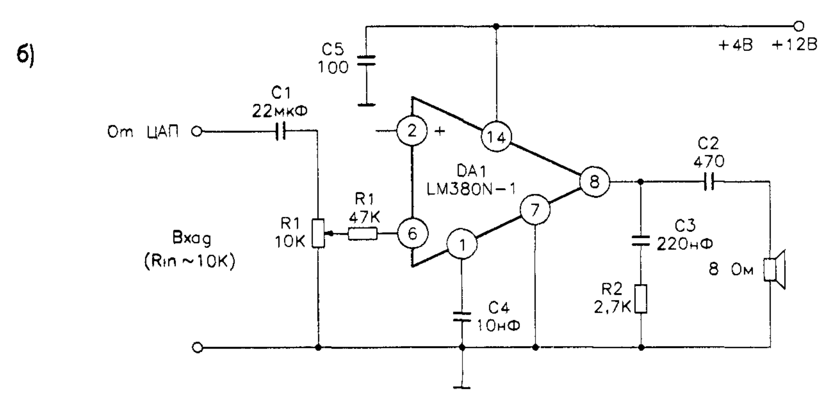


Рис. 5.24. Схемы низкочастотных усилителей звука: а – на TBA820M; б – на LM380N-1

Микросхема LM380N-1 (рис. 5.24б) – это другой *усилитель низкой частоты* с фиксируемым коэффициентом усиления, равным 50. С помощью внешних элементов усиление можно увеличить до 200. Микросхема имеет ограничение выходного тока и термозащиту. Напряжение питания должно быть от 4 до 12 В. Минимальное сопротивление нагрузки 8 Ом. Уровень входного сигнала регулируется переменным резистором R2.

5.9. Устройства управления дисплеями

В разделе приводится описание принципиальных электрических схем и электронных приборов, предназначенных для управления светодиодными и жидкокристаллическими дисплеями.

5.9.1. Многоразрядные светодиодные дисплеи со встроенными схемами управления

Микросхема TSM6234T (Three Five System) – это *четырехразрядный зеленый светодиодный дисплей* шириной 0,3 дюйма со встроенным последовательным входом (рис. 5.25). Потребляемый каждым сегментом ток равен 2,0 мА. Ток, необходимый светодиодам, определяется внешним резистором и обычно в 25 раз превышает ток, протекающий через вывод управления яркостью (контакт 7).

Между выводом управления яркостью и «землей» должен быть включен конденсатор 0,1 мкФ. Для работы дисплея нужно два напряжения питания: Vdd и Vled. Напряжение Vdd предназначено для питания внутренней схемы управления и может меняться от 4,75 до 12 В. Потребляемый ток равен 7 мА для напряжения 12 В. Напряжение Vled обычно составляет 5 В и служит для питания светодиодов дисплея.

Последовательная передача данных осуществляется по трем ТТЛ совместимым линиям: «вход данных» (контакт 4), $\overline{\text{ENABLE}}$ (контакт 3) и CLOCK (контакт 5). На рис. 5.26 изображены временные диаграммы загрузки данных в дисплей.

Формат передачи данных состоит из стартовой единицы и 35 бит данных. По каждому положительному фронту тактового импульса биты данных последовательно записываются во входной сдвиговый регистр. Чтобы открыть вход, надо подать на контакт 3 ($\overline{\text{ENABLE}}$) сигнал низкого уровня. При прохождении 36-го фронта тактового импульса генерируется сигнал загрузки, который переводит 35 бит данных из регистра сдвига в буфер-защелку. Во время прохождения следующего фронта формируется сигнал «сброс», который очищает регистр сдвига. При включении питания генерируется сигнал «сброс при включении», который очищает все регистры сдвига и буфер-защелку. Стартовый бит и тактовый импульс возвращают микросхему в режим загрузки данных. Для очистки дисплея необходимо подать стартовый бит и 35 нулей. Эта процедура также сбрасывает микросхему. Бит 1, следующий сразу за стартом, определяет состояние сегмента A первой цифры, бит 2 – состояние сегмента B первой цифры и т.д.

Функции 35 бит последовательных данных можно определить так:

биты 1–8	сегменты A–DP первой цифры
биты 9–16	сегменты A–DP второй цифры

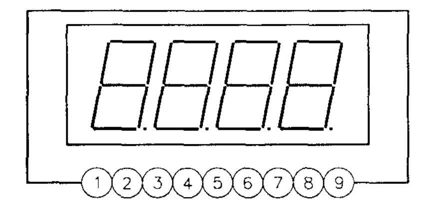


Рис. 5.25. Расположение выводов и внутренняя блок-схема TSM6234

биты 17–24	сегменты A–DP третьей цифры
биты 25–32	сегменты A–DP четвертой цифры

Схема с использованием экспериментальной платы последовательного порта изображена на рис. 5.27. Выводы «вход данных» и CLOCK подключены к контактам RTS и DTR на плате. Контакт 3 ($\overline{\text{ENABLE}}$) соединен с «землей» для того, чтобы вход данных был постоянно открыт.

Текст программы TSM6234.PAS

```
Program TSM6234,
(*Программа управления четырехразрядным светодиодным дисплеем TSM6234 *)
(*Соединен с экспериментальной платой последовательного порта,
  DATA соединен с RTS,
  CLOCK соединен с DTR *)
```

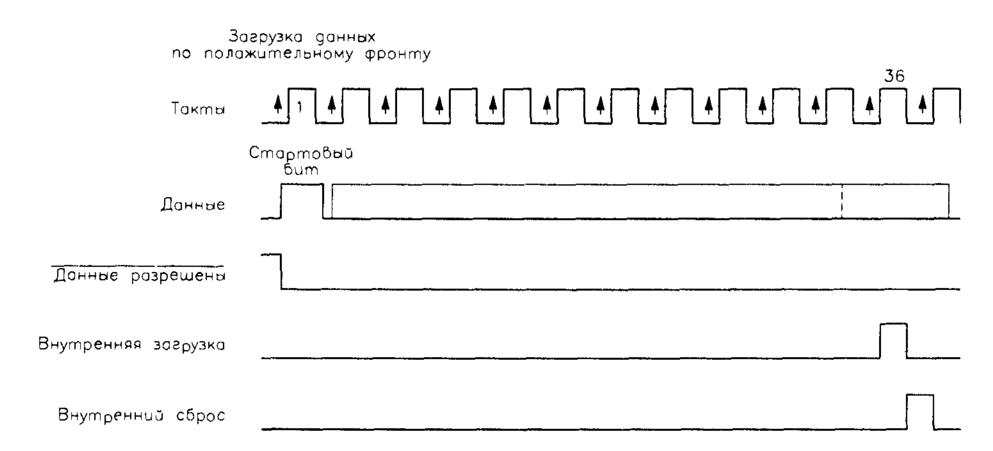


Рис. 5.26. Временные диаграммы светодиодного дисплея TSM6234

```
uses
  crt,dos,
{$I c \ioexp\tplib1.pas}
procedure start,
(*Загрузка стартового бита  1 - по входу данных, переход 0-1 по входу тактов.*)
begin
  write_modem_status(RS232_address,1,0);
  write_modem_status(RS232_address,1,1);
  write_modem_status(RS232_address,1,0);
end,
procedure load_bit(bitx byte),
(*Загрузка бита данных, bitx *)
begin
  write_modem_status(RS232_address,bitx,0),
  write_modem_status(RS232_address,bitx,1);
  write_modem_status(RS232_address,bitx,0),
end,
function segment_data(charx:char)·byte;
(*charx - отображаемый символ (0-9, A-F).*)
(*Функция вычисляет двоичные значения для сегментов дисплея.*)
(*Двоичные данные: сегменты a,b,c,d,e,f,dp=DB0,DB1,DB2,...,DB7.*)
begin
  if charx= 0' then segment_data.=$3F;
  if charx='1' then segment_data:=$06;
  if charx='2  then segment_data·=$5B;
  if charx='3' then segment_data:=$4F,
  if charx='4' then segment_data =$66;
  if charx='5' then segment_data =$6D,
  if charx='6  then segment_data·=$7D;
```

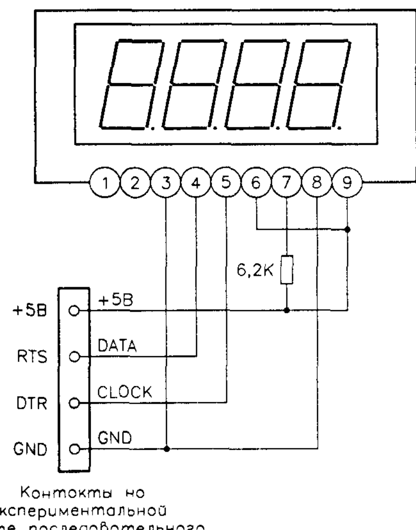


Рис. 5.27. Схема управления светодиодным дисплеем TSM6234

```
  if charx='7' then segment_data:=$07,
  if charx='8' then segment_data.=$7F;
  if charx='9' then segment_data:=$6F;
  if upcase(charx)='A' then segment_data =$77;
  if upcase(charx)='B' then segment_data·=$7C,
  if upcase(charx)='C' then segment_data =$39,
  if upcase(charx)='D' then segment_data =$5E;
  if upcase(charx)='E' then segment_data:=$79,
  if upcase(charx)='F' then segment_data·=$71;
  if upcase(charx)=  ' then segment_data.=$00;
end;

procedure load_digits(strx·string);
(*Загрузка 34 бит данных *)
(*Всего тактовых импульсов: 35 *)
var
  i,j.integer,
  bitvalue:byte;
begin
  for j:=1 to 4 do
  for i:=1 to 8 do
     begin
        load_bit(round(segment_data(srtx[j]) and bit_weight(i)/bit_weight(i)));
     end,
  for i:=1 to 2 do load_bit(0);
  load_bit(0);
end,
```

```
procedure loaddata_test;
var
  i:integer;
  digit_string:string;
begin
  write_transmit_buffer(RS232_address,0);
  repeat
     clrscr;
     start;
     writeln('Input Q or q to quit the program');
     write('Input four digits (0,1,2...9,a,b...f): '); readln(digit_string);
     load_digits(digit_string);
  until upcase(digit_string[1])='Q';
end;

(*Главная программа.*)
begin
  COM_address;
  Loaddata_test;
end.
```

5.9.2. Растровые светодиодные дисплеи со встроенными схемами управления

Микросхемы RS590-935 и RS590-941 – это двухдюймовые *растровые светодиодные дисплеи* с разрешением 5×7 пикселов, которые можно использовать для отображения больших символов (рис. 5.28). Встроенная КМОП логика управления содержит модуль памяти, генератор символов в коде ASCII, мультиплексор и схему управления. Это позволяет дисплею отображать 96 символов ASCII без дополнительных схем управления.

Линии ввода/вывода совместимы с уровнем ТТЛ и позволяют соединять несколько дисплеев. Когда все светодиоды горят, напряжение питания равно 5 В. Устройство потребляет ток 80 мА. Яркость можно уменьшить в два или в четыре раза. Контакт 3 – это вход разрешения микросхемы. Для записи данных в дисплей на него необходимо подать низкий уровень, при этом на входах 8–14 уже должны присутствовать параллельные данные D0 – D6. Затем на вход $\overline{WR}$ (контакт 4) подается стробирующий импульс, по положительному фронту которого информация загружается в дисплей. Входные данные (D0 – D6) определяют один из 96 символов, представленных в коде ASCII. Каждый символ отображается до тех пор, пока не будет заменен другим. Яркость светодиодов регулируется входами BL0 и BL1 (контакты 5 и 6):

- BL0 = 0, BL1 = 0 – дисплей не светится;
- BL0 = 0, BL1 = 1 – 1/4 яркости;
- BL0 = 1, BL1 = 0 – 1/2 яркости;
- BL0 = 1, BL1 = 1 – максимальная яркость.

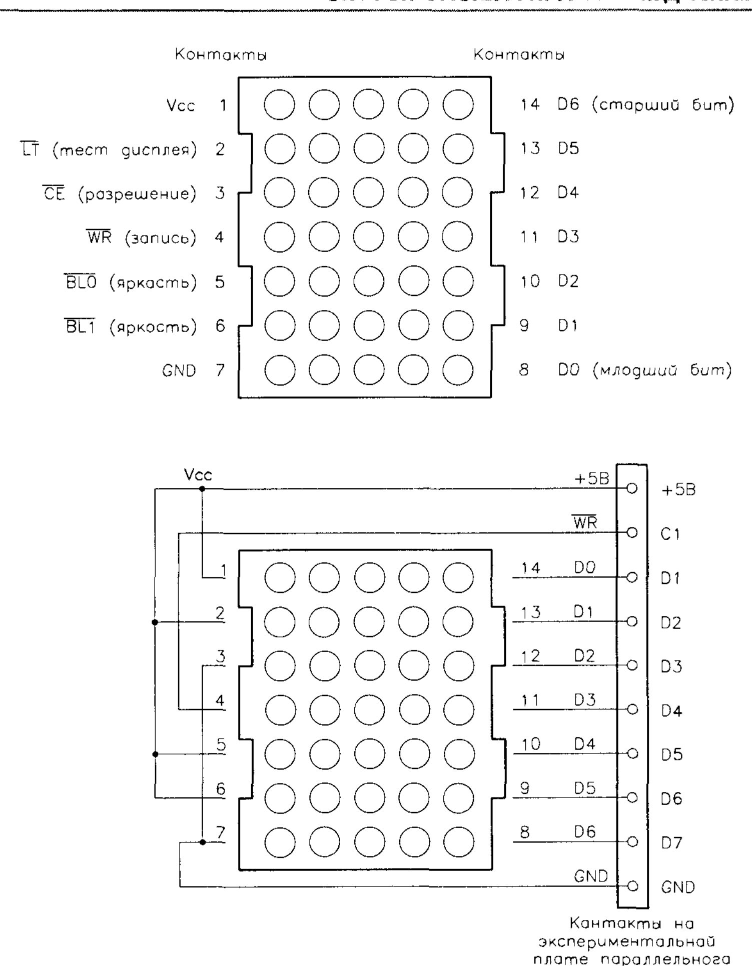


Рис. 5.28. Назначение выводов и схема с использованием растровых дисплеев RS590-935 и RS590-941

Контакт 2 – это вход теста дисплея ($\overline{\text{LT}}$). Если на него подать низкий уровень, то яркость свечения точек растра уменьшится в четыре раза по сравнению с максимальной. Тест не влияет на ранее отображенный символ.

На рис. 5.28 представлена экспериментальная схема. Входы дисплея D0 – D6 соединены с контактами D1 – D7 экспериментальной платы параллельного порта. Линия $\overline{\text{WR}}$ соединена с контактом C1. На контакты 2, 5 и 6 подан высокий уровень. Поскольку набор символов дисплея представлен в коде ASCII, который также используется в компьютере, то любой символ можно трансформировать в двоичный код с помощью процедуры TP6 `ORD(char)`.

Текст программы RS590935.PAS

```
Program LED590935_dot_matrix_display,
(*Программа управления двухдюймовым микропроцессорным растровым дисплеем *)
(*Дисплей соединен с экспериментальной платой параллельного порта,
D1 - D7 соединены с выводами D0 - D6 дисплея,
C1 соединен с выводом WR дисплея *)
uses
  crt,dos,
var
  character char

{$I c \ioexp\tplib1 pas}

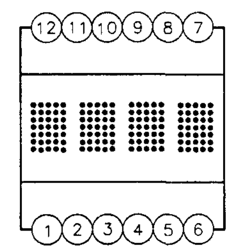
procedure display(character char)
(*Отображение символа на дисплее *)
var
  i integer,
begin
  write_data_port(P_address,ord(character)),    (*Вывод двоичных данных *)
  writeln( Output value for the character is  ,ord(character)),
  write_control_port(P_address,1),              (*WR=1 *)
  write_control_port(P_address,0),              (*WR=0 *)
  write_control_port(P_address 1)               (*WR=1, данные загружены *)
end,

begin
  centronic_address,
  repeat
     write( Input the character to be displayed ), readln(character),
     display(character)
  until ord(character)=13,
end
```

5.9.3. Многоразрядные светодиодные растровые дисплеи со встроенными схемами управления

Микросхема DLR1414 (Siemens, RS589-301) – это *четырехразрядный растровый дисплей* с разрешением каждого разряда 5×7 пикселов и встроенной КМОП схемой управления (рис. 5.29). Встроенная логика содержит модуль памяти, генератор символов в коде ASCII, мультиплексор и схему управления. Входы совместимы с уровнями ТТЛ. Напряжение питания 5 В. Устройство наращиваемое, что позволяет строить систему отображения из нескольких одиночных модулей. Размер отображаемого символа 3,66 мм. Во время работы на входы данных (D0 – D6) и адреса (A0 и A1) необходимо подать информацию. После поступления на вход $\overline{WR}$ (контакт 3) стробирующего импульса происходит загрузка информации. Нулевой разряд располагается справа, при этом на входах A0 и A1 должен быть сигнал низкого уровня.

На рис. 5.30 приведена экспериментальная схема управления дисплеем. Выводы D0 – D6 дисплея подключены к контактам D1 – D7 на экспериментальной плате параллельного порта. $\overline{WR}$ соединен с контактом C1 на плате, линии A0



Назначение контактов

1 D5 – вход данных
2 D4 – вход данных
3 $\overline{\text{WRITE}}$
4 A1 – выбор цифры
5 A0 – выбор цифры
6 Vcc
7 Общий
8 D0 – вход данных (младший байт)
9 D1 – вход данных
10 D2 – вход данных
11 D3 – вход данных
12 D6 – вход данных (старший байт)

Строки 0 6
Дисплей
Столбцы 0 19
Управление строками
3 2 1 0
Логика управления и синхронизации
OSC
Счетчик до 128
Счетчик до 7
Управление загрузкой
D6 12
D5 1
D4 2
D3 11
D2 10
D1 9
D0 8
ОЗУ
Семиразрядный ASCII-код
Дешифратор строк
Логика управления светодиодами дисплея
Триггеры-защелки
Дешифратор столбцов
ПЗУ ASCII-символов 128x35 бит (4480 бит)
$\overline{\text{WR}}$ 3
A0 5
A1 4
Дешифратор адреса записи
Данные в столбцах

Рис. 5.29. Расположение выводов и внутренняя блок-схема микросхемы DLR1414

и A1 – с C2 и C3. Так как наборы символов дисплея и компьютера совпадают, перед подачей на вход микросхемы символ достаточно преобразовать в двоичный код посредством процедуры TP6 ORD(char).

Текст программы 1414.PAS

```
Program RS1414_dot_matrix_display,
(*Программа управления четырехразрядным растровым дисплеем *)
(*Дисплей соединен с экспериментальной платой параллельного порта,
   D1 - D7 соединены с выводами D0 - D6 дисплея,
   C1 соединен с WR,
   A0 и A1 соединены с контактами C2 и C3 платы *)
```

Назначение контактов

1	D5
2	D4
3	$\overline{WR}$
4	A1
5	A0
6	Vcc
7	GND
8	D0
9	D1
10	D2
11	D3
12	D6

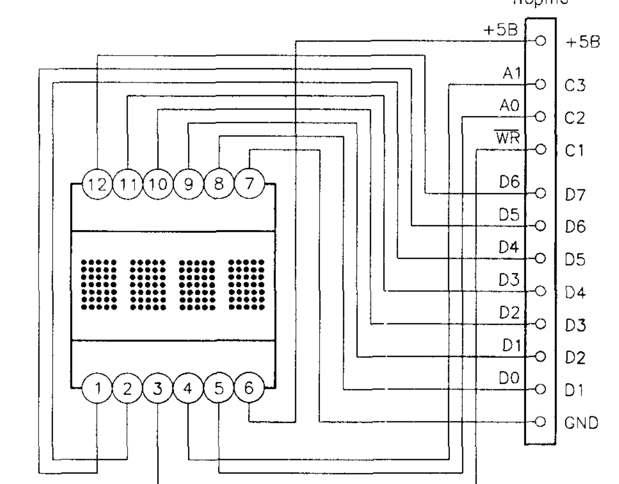


Рис. 5.30. Схема управления дисплеем DLR1414

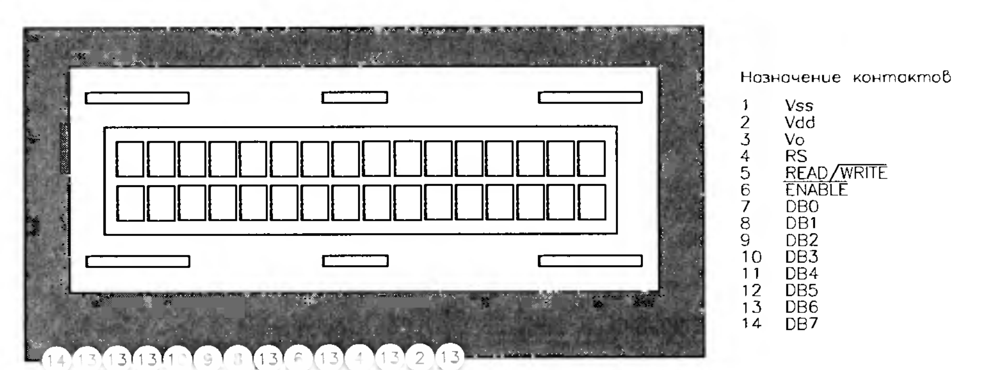
```
uses
  crt,dos,
var
  ch1,ch2,ch3,ch4 char,
{$I c \ioexp\tplib1 pas}
procedure display(character char,digit byte),
(*Отображение символа на дисплее *)
var
  i integer,
begin
  write_data_port(P_address,ord(character)),    (*Вывод двоичных данных *)
  writeln( Output value for the character is  ,ord(character))
  write_control_port(P_address,1+digit*2),      (*WR=1 *)
  write_control_port(P_address,0+digit*2),      (*WR=0 *)
  write_control_port(P_address,1+digit*2),      (*WR=1, данные зафиксированы *)
end,
procedure load_digits(ch1,ch2,ch3,ch4 char),
(*Отображение четырех разрядов *)
begin
  display(ch1,3),
  display(ch2,2),
  display(ch3,1),
  display(ch4,0),
end,
begin
  centronic_address
  repeat
```

```
      write( Input four characters to be displayed (from left to right) ),
      readln(ch1,ch2,ch3,ch4),
      load_digits(cg1,ch2,ch3,ch4),
   until ord(ch1)=13,
end
```

5.9.4. Жидкокристаллические растровые дисплейные модули

Микросхема HD44780 (Hitachi) – это *жидкокристаллический дисплейный модуль*, который отображает две строки символов (рис. 5.31). Каждый символ может использовать 5×10 точек матрицы, из которых 5×7 разрешается выбирать программно. Дисплей имеет набор инструкций по выполнению различных функций.

Для ввода/вывода применяется восьмиразрядная двунаправленная шина данных. Кроме того, при организации ввода/вывода необходимо управлять следующими



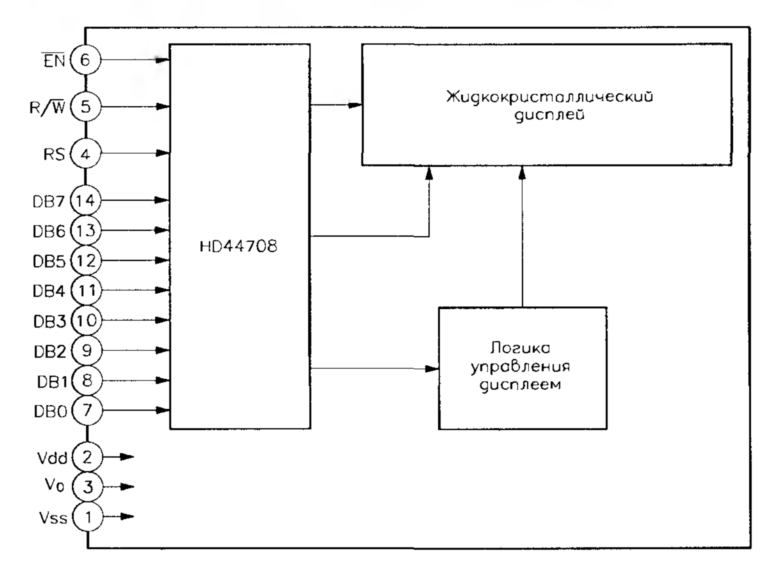


Рис. 5.31. Назначение выводов и внутренняя блок-схема ЖК дисплея

входами микросхемы: контакт 4 (выбор регистра, RS), контакт 5 (чтение/запись, $R/\overline{W}$) и контакт 6 (разрешение, $\overline{ENABLE}$).

Линия RS отображает то, что загружается в модуль дисплея – инструкции или данные (RS = 0: передаются инструкции; RS = 1: передаются данные). Линия $R/\overline{W}$ показывает, какая операция проводится в данный момент – чтение или запись ($R/\overline{W}$ = 1: чтение; $R/\overline{W}$ = 0: запись). Чтением и записью данных управляет линия $\overline{ENABLE}$. При загрузке данных информация подается на входы RS, $R/\overline{W}$, DB0 – DB7, а затем по отрицательному фронту на входе $\overline{ENABLE}$ (рис. 5.32) загружается в дисплей. Набор основных инструкций приведен на рис. 5.33 (полный список можно получить у производителя).

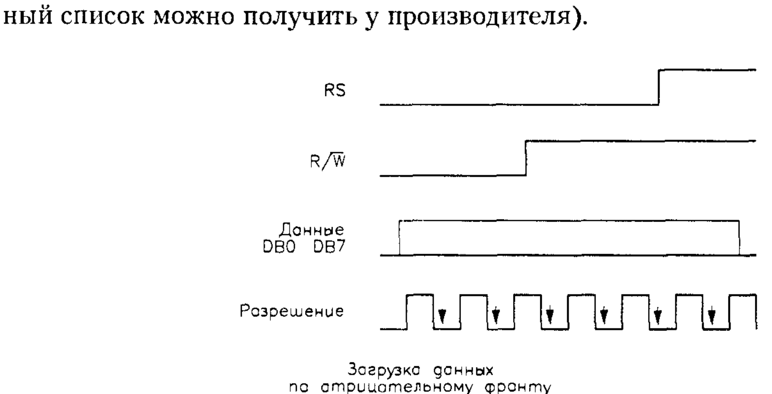


Рис. 5.32. Временные диаграммы работы растрового ЖК дисплея

Схема с использованием дисплея изображена на рис. 5.34. Здесь используется режим работы с данными. Входы D0 – D7 подключены к контактам D1 – D8 экспериментальной платы параллельного порта. Вход $\overline{ENABLE}$ соединен с контактом C1 на плате, входы RS и $R/\overline{W}$ – с C2 и C3. Поскольку наборы символов дисплея и компьютера совпадают, перед подачей на вход микросхемы символ достаточно преобразовать в двоичный код при помощи процедуры TP6 ORD(char).

Текст программы LM016L.PAS

```
Program LM016L_dot_matrix_display,
(*Программа управления жидкокристаллическим дисплеем.
  Дисплей соединен с экспериментальной платой параллельного порта,
  D1-D8 соединены с выводами D0-D7 дисплея,
  C1 соединен с EN дисплея
  RS и R/W соединены с C2 и C3.*)
uses
  crt,dos,
var
  sen1,sen2:string[16];

{$I c:\ioexp\tplib1.pas}

procedure control(command:byte);
(*Ввод управляющего слова в модуль дисплея *)
```

Инструкции	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Описание
Очистить дисплей	0	0	0	0	0	0	0	0	0	1	Очистка дисплея возврат курсора в начальную позицию
Возврат в начало	0	0	0	0	0	0	0	0	1	0	Возврат курсора в начальную позицию, ОЗУ не изменяется
Установка режима	0	0	0	0	0	0	0	1	I/D	S	Установка режима дисплея
Включение/выключение дисплея	0	0	0	0	0	0	1	D	C	B	Включение/выключение D – дисплея, C – курсора, B – мигающего курсора
Сдвиг курсора	0	0	0	0	0	1	S/C	R/L	0	0	Перемещение курсора и скроллинг, ОЗУ не изменяется
Установка функций	0	0	0	0	1	DL	N	F	0	0	Настройка интерфейса ввода/вывода
Установка CGRAM-адреса	0	0	0	1	ACG						Установка CGRAM-адреса, после этого передача или прием данных
Установка DDRAM-адреса	0	0	1	ADD							Установка DDRAM-адреса, после этого передача или прием данных
Чтение адреса и флага занятости	0	1	BF	AC							Чтение флага занятости и счетчика адреса
Запись данных	1	0	Запись данных								Запись в ОЗУ
Чтение данных	1	1	Чтение данных								Чтение из ОЗУ
	I/D=1 увеличение, I/D=0 уменьшение S=1 относительный скролинг S/C=1 скроллинг, S/C=0 перемещение курсора R/L=1 сдвиг вправо, R/L=0 сдвиг влево DL=1 8 бит, DL=0 4 бита N=1 2 строки, N=0 1 строка F=1 матрица 5x10, F=0 матрица 5x7 BF=1 работа по внутренним инструкциям BF=0 прием внешних инструкций										DDRAM вывод содержимого ОЗУ CGRAM ОЗУ генератора символов ACG адрес CGRAM ADD адрес DDRAM, адрес курсора AC счетчик адреса, используемый для DDRAM и CGRAM

Рис. 5.33. Набор инструкций

```
var
  i:integer;
begin
  write_data_port(p_address,command);      (*Ввод двоичных данных *)
  write_control_port(P_address,0+0+0);     (*EN=1, RS=0, R/W=0.*)
  write_control_port(P_address,1+0+0);     (*EN=0, RS=0, R/W=0.*)
  write_control_port(P_address,0+0+0);        (*EN=1, данные зафиксированы.*)
  delay(100);
end,

procedure clear;
(*Сброс всех разрядов.*)
begin
  control(1);
end;

procedure home;
(*Перемещение курсора на исходную позицию.*)
begin
  control(2);
end,

procedure line1;
(*Перемещение курсора на первую строку *)
```

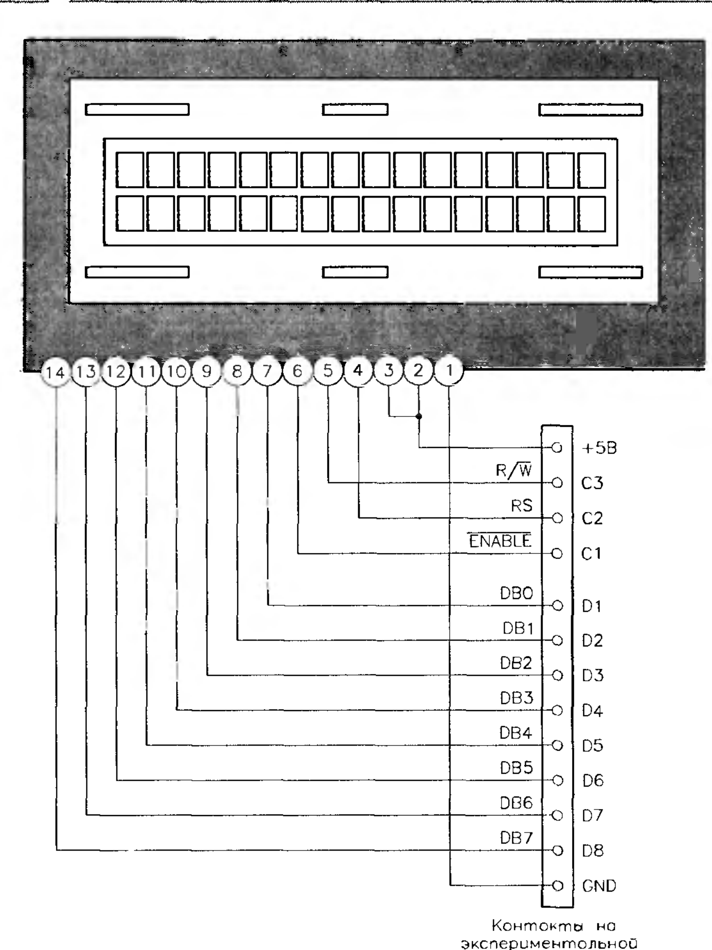


Рис. 5.34. Схема с использованием растрового ЖК дисплея

```
begin
  control(128),
end

procedure line2,
(*Перемещение курсора на вторую строку *)
begin
  control(128+64),
end,

procedure data(ch char),
(*Отображение символа на дисплее *)
var
  i integer,
```

```
begin
  write_data_port(p_address,command),        (*Ввод двоичных данных *)
  write_control_port(P_address,0+2+0),       (*EN=1, RS=2, R/W=0 *)
  write_control_port(P_address,1+2+0),       (*EN=0, RS=2, R/W=0 *)
  write_control_port(P_address,0+2+0),       (*EN=1, данные зафиксированы *)
  delay(10);
end;

procedure initialization,
begin
  control(16+32);      (*Установка функции, 8 бит данных, 2 строки, 5x7 точек *)
  control(16+32),      (*Установка функции, 8 бит данных, 2 строки, 5x7 точек.*)
  control(16+32),      (*Установка функции, 8 бит данных, 2 строки, 5x7 точек *)

  control(16+32+8+4),  (*Установка функции, 8 бит данных, 2 строки, 5x10 точек *)
  control(8+4+2+1),    (*8=бит управления, 4=включение разрядов, 2=включение курсора,
                       1=символ курсора мигает *)
  clear,
  control(4+2+0);      (*Установка режима ввода, 4=бит управления, 2=увеличение,
                       0=курсор без сдвига *)
end,

procedure test;
(*Простая тестирующая программа *)
var
  i integer;
begin
  sen1.='                ';
  sen2:='                ',
  write('Input the first sentence (max 16 char):'),
  readln(sen1);
  write('Input the second sentence (max 16 char) ');
  readln(sen2);

  (*Отображение первой строки.*)
  line1;
  for i:=1 to 16 do
  begin
  data(sen1[i]);

  (*Отображение второй строки.*)
  line2;
  for i:=1 to 16 do
      begin
          data(sen2[i]);
          delay(10);
      end,
end,

begin
  centronic_address;
  initialization;
  test,
end
```

5.10. Устройства управления мускульными кабелями

Мускульные кабели изготавливаются из специальных сплавов, которые при нагреве выше определенного уровня способны изменять форму. Нагрев может быть обусловлен протеканием через кабель электрического тока. При изменении размеров мускульные кабели, работая плавно и бесшумно, перемещают грузы весом в тысячи раз больше их собственного веса.

Одним из наиболее распространенных материалов, используемых для изготовления мускульных кабелей, является флексинол. Флексиноловые провода бывают разных диаметров: 25, 50, 100, 150 и 250 мкм. Рассмотрим кабель диаметром 100 мкм и линейным сопротивлением 150 Ом/м. Рекомендуемый ток, при котором материал еще не деформируется, равен 180 мА. Такой трос способен в течение длительного времени удерживать груз весом 150 г без потери формы. Количество циклов сжатия–растяжения достигает 30 в минуту. Кабель начинает деформироваться при температуре 68 °C, а при температуре 78 °C становится на 8% короче, чем в охлажденном состоянии.

Для управления кабелем допустимо использовать любые транзисторные или релейные устройства коммутации (см. рис. 5.2, 5.6 и 5.7). При настройке протекающего через провод тока может потребоваться нагрузочный резистор, включенный последовательно.

На рис. 5.35 изображена схема шагового двигателя. Здесь мускульный кабель применяется для генерации вращательных движений под контролем компьютера. Схема управляется экспериментальной платой параллельного порта.

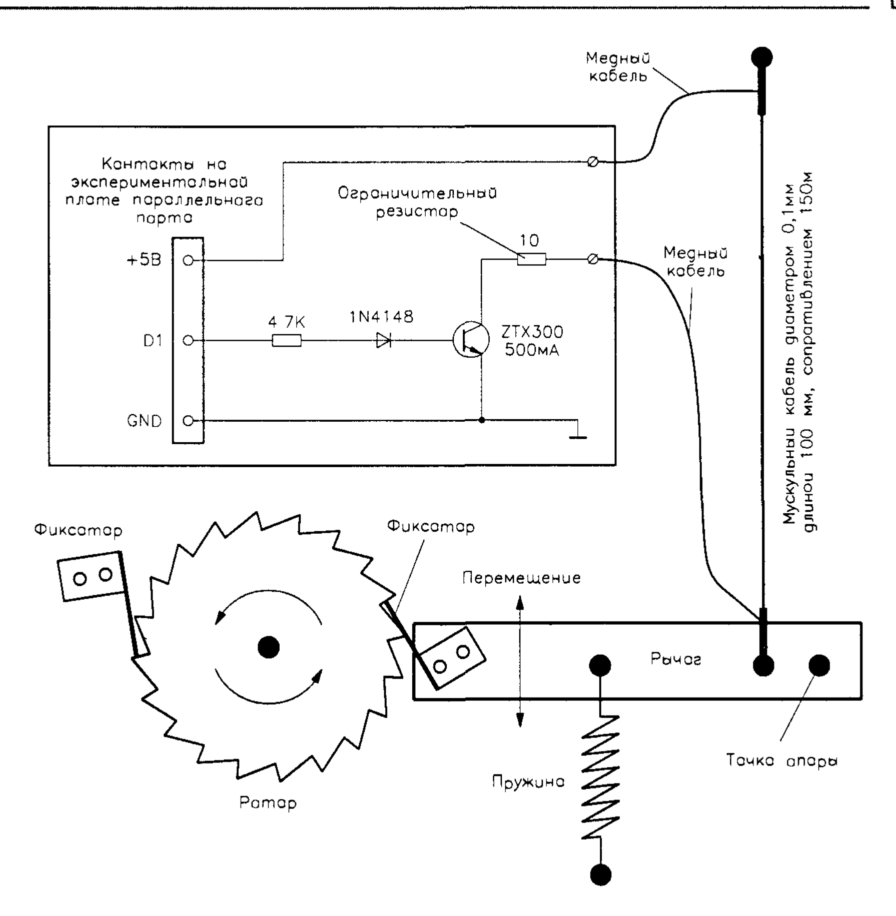


Рис. 5.35. Шаговый двигатель на мускульном кабеле, управляемый параллельным портом ПК

6. ИЗМЕРЕНИЕ АНАЛОГОВЫХ ВЕЛИЧИН

Глава посвящена описанию электронных приборов, которые предназначены для измерения аналоговых величин, таких как напряжение, температура, влажность и т.п. Дается информация о различных способах преобразования аналоговых величин в цифровую форму. Приводятся принципиальные электрические схемы и программы управления, способные превратить компьютер в измерительный комплекс.

6.1. Аналого-цифровые преобразователи

Аналого-цифровые преобразователи (АЦП) формируют на выходе двоичный код, прямо пропорциональный входному напряжению. Это одно из основных устройств, позволяющее передавать аналоговые сигналы в компьютер. Существует три основных типа АЦП: параллельные, последовательного приближения и с двойным интегрированием. Параллельные АЦП имеют наибольшую скорость преобразования; у АЦП с двойным интегрированием максимальная точность преобразования, но они менее быстрые. В качестве компромисса между скоростью и точностью выступают АЦП последовательного приближения. Один из основных параметров, влияющих на точность преобразования, – это *шаг квантования* (Vlsb), равный приращению напряжения, которое необходимо для изменения кода в младшем разряде. Существует два типа интерфейса ввода/вывода между АЦП и внешними схемами: параллельный и последовательный. При параллельном интерфейсе выходные данные снимаются через параллельную шину, имеющую несколько линий данных, при последовательном данные считываются по одной линии. В обоих случаях необходимы линии управления.

6.1.1. АЦП с параллельным интерфейсом ввода/вывода

В данном разделе описаны различные способы преобразования аналогового напряжения в цифровую форму и представлены соответствующие микросхемы.

Параллельный АЦП CA3306

Принцип параллельного АЦП состоит в том, что входной сигнал сравнивается с опорными напряжениями (рис. 6.1).

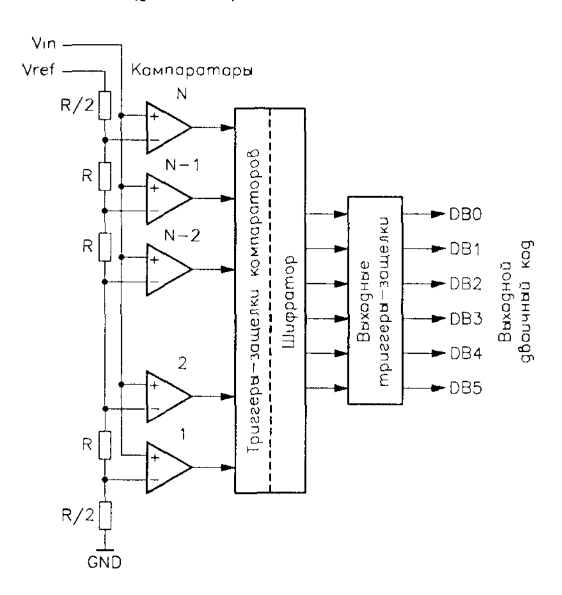


Рис. 6.1. Принцип построения шестиразрядного параллельного АЦП

Опорные напряжения формируются с помощью включенных последовательно резисторов, образующих делитель. Точность преобразования равна шагу квантования в середине диапазона и половине шага по краям. Напряжение, подаваемое на нижний компаратор, составляет 0,5Vlsb, на следующий компаратор – 1,5Vlsb. Ни один из компараторов не сработает, когда напряжение на входе равно нулю. Если уровень входного напряжения находится между фиксированными значениями опорного напряжения, то срабатывает младший компаратор. Например, если значение входного напряжения находится между 0,5Vlsb и 1,5Vlsb, то выбирается уровень 0,5Vlsb. Формируемый компараторами код преобразуется шифратором в параллельный двоичный код соответствующей разрядности. С увеличением разрядности АЦП экспоненциально растет количество компараторов: так, n-разрядный преобразователь должен содержать 2^n компараторов.

Микросхема CA3306CE (Harris Semiconductor, RS648-652) имеет частоту преобразования 15 МГц, точность преобразования 6 бит. Она содержит 64 компаратора, работает при напряжении питания от 3,0 до 7,5 В, потребляемая мощность равна 50 мВт (рис. 6.2).

Преобразованный двоичный код снимается с выходов B1 – B6. Контакт 2 служит для индикации переполнения. Также имеется два разрешающих входа: CE2

Рис. 6.2. Назначение выводов и схема с использованием АЦП CA3306

(контакт 5) и $\overline{CE1}$ (контакт 6). Если CE2 = 1 и $\overline{CE1}$ = 0, то выход открыт для считывания данных, в противном случае выходы имеют большое сопротивление.

Контакт 7 – это тактовый вход, контакт 8 управляет последовательными операциями преобразования· если на нем высокий уровень, то по положительному фронту тактового импульса происходит преобразование. При прохождении отрицательного фронта трансформированные данные загружаются в триггеры-защелки компараторов. При низком уровне тактового импульса шифратор превращает данные в параллельный двоичный код. При следующем положительном фронте данные загружаются в выходные триггеры и появляются на выходах микросхемы. Этот же фронт инициирует новый цикл преобразования. Таким образом, на

выходе присутствуют данные, соответствующие предыдущему тактовому импульсу. Vref– (контакт 10) и Vref+ (контакт 9) – это входы опорного напряжения. Vref– соединяется с «землей», а на Vref+ подается напряжение от источника питания.

ИС, приведенная на рис. 6.2, соединена с экспериментальной платой параллельного порта. Данная схема использует буфер 74LS241, который позволяет считывать шестиразрядный двоичный код с помощью четырех линий. Контакт C1 экспериментальной платы соединен с тактовым входом микросхемы CA3306. Для считывания предыдущего значения и для запуска нового преобразования на контакт C1 необходимо подать положительный импульс; при этом на выходе появится ранее преобразованное значение. Микросхема 74LS241 делит шесть бит данных на две части: два старших и четыре младших бита.

Считыванием информации в компьютер управляет линия выбора данных (DSL), которая подключена к контакту C2.

Текст программы CA3306.PAS

```
Program centronic_CA3306;
(*Программа управления АЦП CA3306.
  АЦП соединен с экспериментальной платой параллельного порта.*)
uses
  crt,graph;
var
  byte_high,byte_low,truebyte:byte;
(*Подключение двух библиотек: TPLIB1 и TPLIB2.*)
{$I c:\ioexp\tplib1.pas}
{$I c:\ioexp\tplib2.pas}
function voltage:real;
(*Считывание входного аналогового напряжения.*)
(*Процедуры write_control_port и read_status_port находятся в библиотеке TPLIB1.*)
begin
(*Начало преобразования.*)
  write_control_port(P_address,0);        (*CLOCK=0, SEL=0.*)
  delay(1);                               (*Кратковременная задержка.*)
  write_control_port(P_address,1);        (*CLOCK=1, SEL=0, предыдущие данные появляются
                                          на шине данных, и начинается новый цикл
                                          преобразования.*)
  write_control_port(P_address,0);        (*CLOCK=0, SEL=0.*)
  (*Считывание результатов преобразования.*)
  byte_low:=read_status_port(P_address);  (*SEL=0, считывается младший байт.*)
  write_control_port(P_address,2);        (*CLOCK=0, SEL=1.*)
  byte_high:=read_status_port(P_address); (*SEL=1, считывается старший байт.*)
  truebyte:=byte_high*16+byte_low;        (*Младший и старший байты объединяются.*)
  voltage:=truebyte/63*5.00;              (*Преобразование двоичного кода в значение
                                          напряжения. Опорное напряжение равно напряжению
                                          питания и составляет 5 В.*)
  (*B1 - B6, 6 бит, максимальное значение 63.*)
end;
(*Главная программа.*)
begin
  clrscr;
  Centronic_address;  (*Получение адреса параллельного порта.*)
```

```
  repeat
     gotoxy(20,10); write('Voltage at the input:',voltage 5:2,'[V]');
     delay(1000);
  until keypressed;
end.
```

Экспериментальная схема упрощается, если параллельный порт двунаправленный. Шесть бит данных могут быть считаны компьютером за один прием, что повышает скорость преобразования. Описанная схема позволяет получить скорость преобразования порядка нескольких сотен килогерц из-за слишком малой скорости передачи данных между микросхемой CA3306 и ПК. Для увеличения скорости преобразования можно временно записывать информацию с выходов CA3306 в буферную память, а при ее заполнении данные будут считываться в компьютер.

АЦП последовательного приближения ZN449

АЦП последовательного приближения содержит ЦАП, компаратор и регистр последовательного приближения (рис. 6.3). Перед началом преобразования в разряды этого регистра записываются нули, что обеспечивает формирование нулевого напряжения на выходе ЦАП. Затем каждый разряд ЦАП, начиная со старшего, последовательно устанавливается в единицу. Одновременно выходное напряжение с ЦАП подается на компаратор и сравнивается с измеряемым. Если выходное напряжение ЦАП больше, то значение проверяемого разряда возвращается в ноль, если меньше, то не изменяется. При такой проверке каждого разряда производится полный цикл преобразования. Для n-разрядного преобразователя требуется n тактов на одно преобразование.

Микросхемы ZN449 или ZN448 (GEC Plessey, RS301-729) – это восьмиразрядные АЦП последовательного приближения (рис. 6.4). Минимальное время

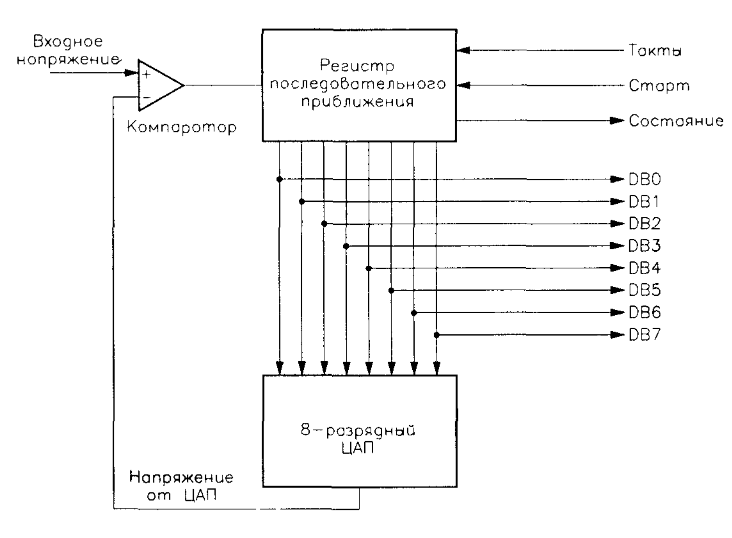


Рис. 6.3. Восьмиразрядный АЦП последовательного приближения

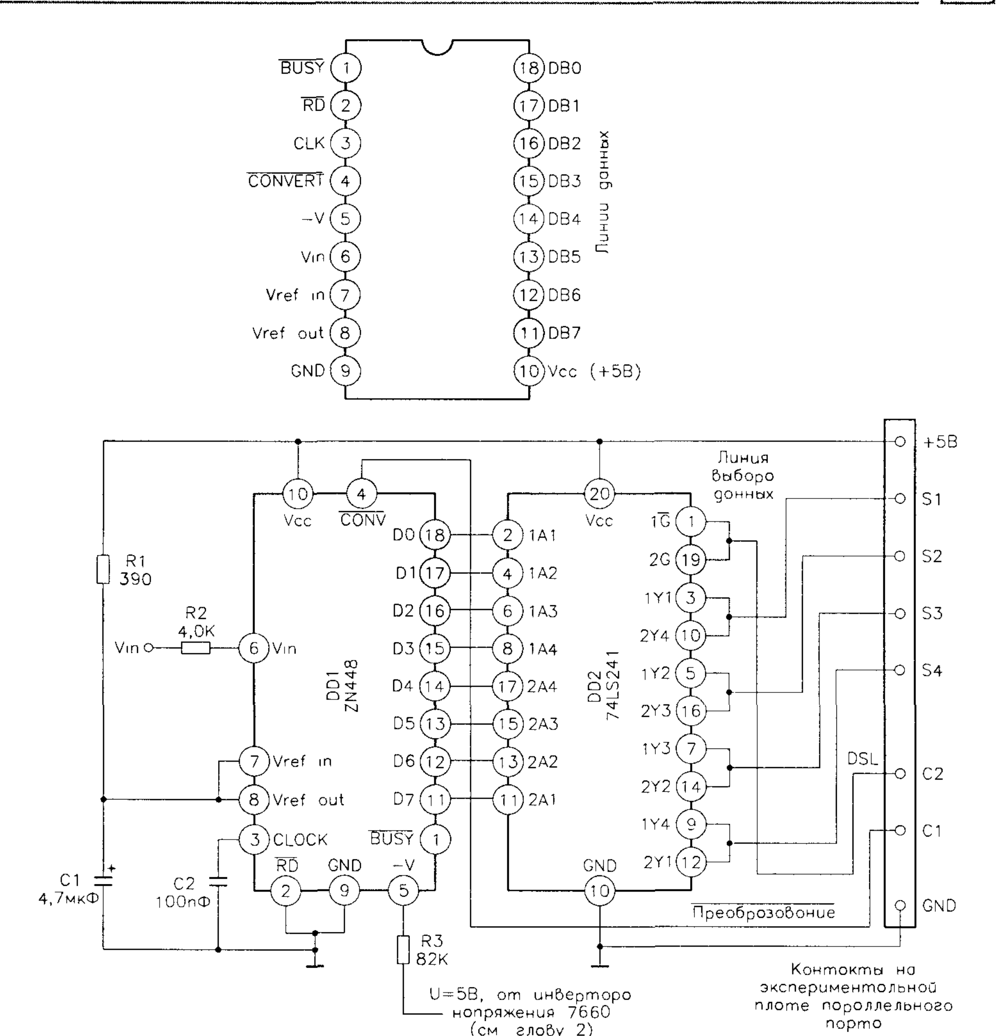


Рис. 6.4. Назначение выводов и экспериментальная схема для АЦП ZN449

преобразования равно 9 мкс. Имеется встроенный тактовый генератор и источник опорного напряжения 2,5 В.

Если на контакт 4 (преобразование, $\overline{\text{CONVERT}}$) подается сигнал низкого уровня, АЦП начинает преобразование. При этом на контакте 1 (занят, $\overline{\text{BUSY}}$) также устанавливается сигнал низкого уровня. Когда процесс заканчивается, выход $\overline{\text{BUSY}}$ сигнализирует об этом высоким уровнем. $\overline{\text{RD}}$ (контакт 2) – это вход разрешения чтения. Если на него подается низкий уровень, то на выходах DB0 – DB7 (контакты 18–11) присутствует код преобразованного напряжения, в противном случае они имеют высокое сопротивление.

Для запуска внутреннего тактового генератора конденсатор C2 необходимо подсоединить к контактам 3 и 9. Максимальная тактовая частота достигается при

емкости конденсатора 100 пФ. Напряжение питания от –3 до –30 В подается на контакт 5 через резистор, значение которого зависит от напряжения питания: например, при –5 В сопротивление резистора равно 82 кОм.

Контакт 8 – это выход встроенного источника опорного напряжения 2,5 В Vref out. Для правильной работы схемы необходимы резистор R1 и развязывающий конденсатор C1. Выход Vref out соединен с входом Vref in (контакт 7). Измеряемое входное напряжение подается на контакт 6 через резистор 4 кОм. Если на входе 2,5 В, на выходе преобразователя будет число 255 (десятичное). Десятичные значения для остальных напряжений вычисляются по следующей формуле:

$$\text{Десятичное число} = \frac{\text{Входное напряжение}}{2{,}5}\,255.$$

Пример использования микросхемы совместно с экспериментальной платой параллельного порта приведен на рис. 6.4. Вход $\overline{\text{CONVERT}}$ соединен с контактом C1, линия выбора данных 74LS241 – с контактом C2. Данные считываются в компьютер через линии S1 – S4. Напряжение –5 В генерируется схемой преобразователя, представленного в главе 2. Программа управления написана на языке TP6.

Текст программы ZN449.PAS

```
Program centronic_ZN449,
(*Программа управления микросхемой АЦП ZN449 с помощью экспериментальной платы параллельного
порта.*)
uses
  crt,graph;
var
  byte_high,byte_low,truebyte:byte;

(*Подключение двух библиотек: TPLIB1 и TPLIB2.*)
{$I c:\ioexp\TPLIB1.pas}
{$I c:\ioexp\TPLIB2.pas}

function voltage:real;
(*Считывание входного аналогового напряжения.*)
(*Процедуры write_control_port и read_status_port находятся в библиотеке TPLIB1.*)
begin
(*Начало преобразования.*)
  write_control_port(P_address,0);          (*CONVERSION=0, SEL=0.*)
  delay(1);                                 (*Кратковременная задержка: ожидание
                                            окончания преобразования.*)
  write_control_port(P_address,1);          (*CONVERSION=1, SEL=0,
                                            окончание преобразования.*)
  delay(1);                                 (*Кратковременная задержка.*)

  (*Считывание результатов преобразования.*)
  byte_low:=read_status_port(P_address);    (*SEL=0, считывается младший байт *)
  write_control_port(P_address,3);          (*CONVERSION=1, SEL=1.*)
  byte_high:=read_status_port(P_address);   (*SEL=1, считывается старший байт *)
  truebyte:=byte_high*16+byte_low;          (*Младший и старший байты объединяются.*)
  voltage:=truebyte/255*2 50;               (*Преобразование двоичного кода в значение
                                            напряжения. Опорное напряжение равно 2,5 В.*)
end;
```

```
(*Главная программа.*)
begin
  clrscr;
  Centronic_address;  (*Получение адреса параллельного порта.*)
  repeat
     gotoxy(20,10), write('Voltage at the input:',voltage:5.2,'[V]');
     delay(1000);
  until keypressed;
end.
```

12-разрядный АЦП с двойным интегрированием ICL7109

Принцип действия АЦП с двойным интегрированием поясняется на рис. 6.5. Данный метод преобразования использует интегратор и опорное напряжение, полярность которого противоположна измеряемому. В такие АЦП, как правило, встроены формирователи опорного напряжения, однако для прецизионных измерений необходимо применять внешние высококачественные источники опорного напряжения.

Преобразование происходит в два этапа. На первом этапе ключ S1 замкнут, а S2 разомкнут. В течение фиксированного промежутка времени Tint интегрируется входное напряжение, затем ключ S1 размыкается, а S2 замыкается. Начинается второй этап преобразования: на вход интегратора подается опорное напряжение противоположной полярности. Конденсатор интегратора разряжается со скоростью, определяемой значением опорного напряжения. Когда выходное напряжение интегратора становится равным нулю, измеряется период времени Tdeint. Зная два периода времени, величину входного напряжения можно вычислить по следующей формуле:

$$\text{Входное напряжение} = \text{Vref}\,\frac{\text{Tdeint}}{\text{Tint}}.$$

Преимущество АЦП с двойным интегрированием состоит в том, что погрешность преобразования не зависит от погрешности элементов, входящих в интегрирующую цепь (резистор и конденсатор), так как их значения не изменяются в течение периода преобразования. Важно лишь, чтобы постоянной оставалась тактовая частота, которая определяет измеряемые промежутки времени.

Данный метод обеспечивает высокую помехозащищенность. Случайный шум с помощью автоматического шумоподавления можно свести к нулю еще на этапе интегрирования сигнала. Интерференционные помехи с фиксированными частотами

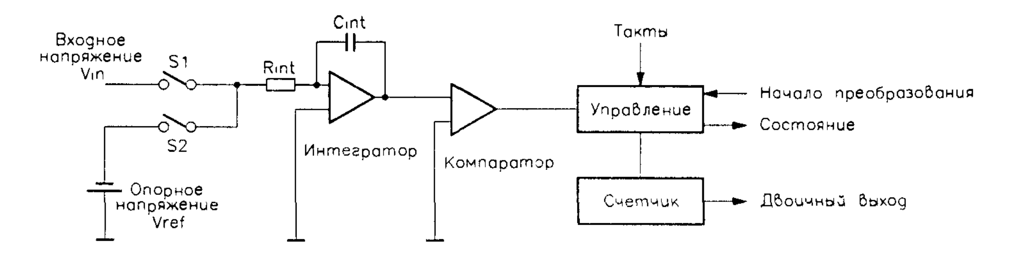


Рис. 6.5. Принцип действия АЦП с двойным интегрировонием

устраняются при правильном подборе периода интегрирования. Чаще всего в подобных АЦП период интегрирования подбирается таким образом, чтобы устранить помехи напряжения сети 50/60 Гц.

Микросхема ICL7109ACPL (Telcom, RS207-0203) – это маломощный 12-разрядный АЦП с двойным интегрированием (рис. 6.6). Входы REF IN+ (контакт 36) и REF IN– (контакт 39) соединены с источниками опорного напряжения. Микросхема имеет встроенный источник опорного напряжения, величина которого на 2,8 В ниже напряжения на входе V+; оно снимается с выхода REF OUT (контакт 29). Входы IN HI (контакт 35) и IN LO (контакт 34) предназначены для подачи входного сигнала.

Рис. 6.6. Назначение выводов и типовое включение микросхемы ICL7109

Аналоговая секция данной микросхемы требует наличия четырех внешних элементов: опорного конденсатора Cref, конденсатора автоматического определения нуля Caz, интегрирующего конденсатора Cint и интегрирующего резистора Rint, значения которых необходимо выбирать в соответствии с рекомендациями изготовителя. Встроенный генератор работает на частоте 3,5795 МГц и позволяет производить 7,5 преобразований в секунду.

Если контакт 26 (RUN/$\overline{\text{HOLD}}$) не используется (внутрисхемно на него подается высокий уровень), то преобразования осуществляются непрерывно. Если на него подать сигнал низкого уровня, преобразование будет производиться только один раз. Его результат появляется на 14 выходах с тремя состояниями: B1 – B12, OR и POL. B1 – B12 служат для вывода измененных данных, OR показывает превышение входного сигнала, а POL – его полярность.

Микросхема имеет два режима выгрузки данных: параллельный режим и режим взаимодействия, которые выбираются с помощью входа MODE (контакт 21). При MODE = 0 реализуется параллельный режим, а при MODE = 1 – режим взаимодействия. В первом случае данные считываются с выходов под управлением инвертированных входов $\overline{\text{CE}}$/LOAD, $\overline{\text{LBEN}}$ и $\overline{\text{HBEN}}$. Для открытия микросхемы на вход $\overline{\text{CE}}$/LOAD необходимо подать низкий уровень. Если на входе $\overline{\text{LBEN}}$ низкий уровень, то на выходах B1 – B8 появляются данные, а если высокий, то выходы имеют высокое сопротивление. Вход $\overline{\text{HBEN}}$ аналогично управляет выходами B9 – B12, OR и POL. В процессе преобразования на выход STATUS (контакт 2) подается высокий уровень. После того как преобразование закончилось, двоичный код загружается в выходные буферы, и выход STATUS сигнализирует об этом низким уровнем. В режиме взаимодействия микросхему ICL7109 можно соединить с UART6402 (подробное описание этого режима приведено в документации производителя).

Когда ИС ICL7109 работает в режиме параллельной выгрузки данных, ее подключают к компьютеру через микросхему 8255PPI. Экспериментальная схема на базе двух мультиплексоров 4051, соединенная с экспериментальной платой параллельного порта, изображена на рис. 6.7. Программа управления написана на языке TP6.

Текст программы 7109PARA.PAS

```
Program ICL_parallel_centronic
(*Программа управления АЦП ICL7109 *)
(*Микросхема ICL7109 работает в режиме параллельной выгрузки данных  Мультиплексоры используются
для передачи данных в компьютер *)
uses
  dos crt,
var
  dummy real,

(*Подключение библиотеки TPLIB1 *)
{$I c \ioexp\TPLIB1 PAS}

function read_voltage real,
(*Получение данных из 7109 и вычисление значения входного напряжения *)
var
  low_byte,high_byte,i byte,
  polarity integer,

begin
  low_byte =0,
  high_byte =0,
  for i =1 to 8 do
```

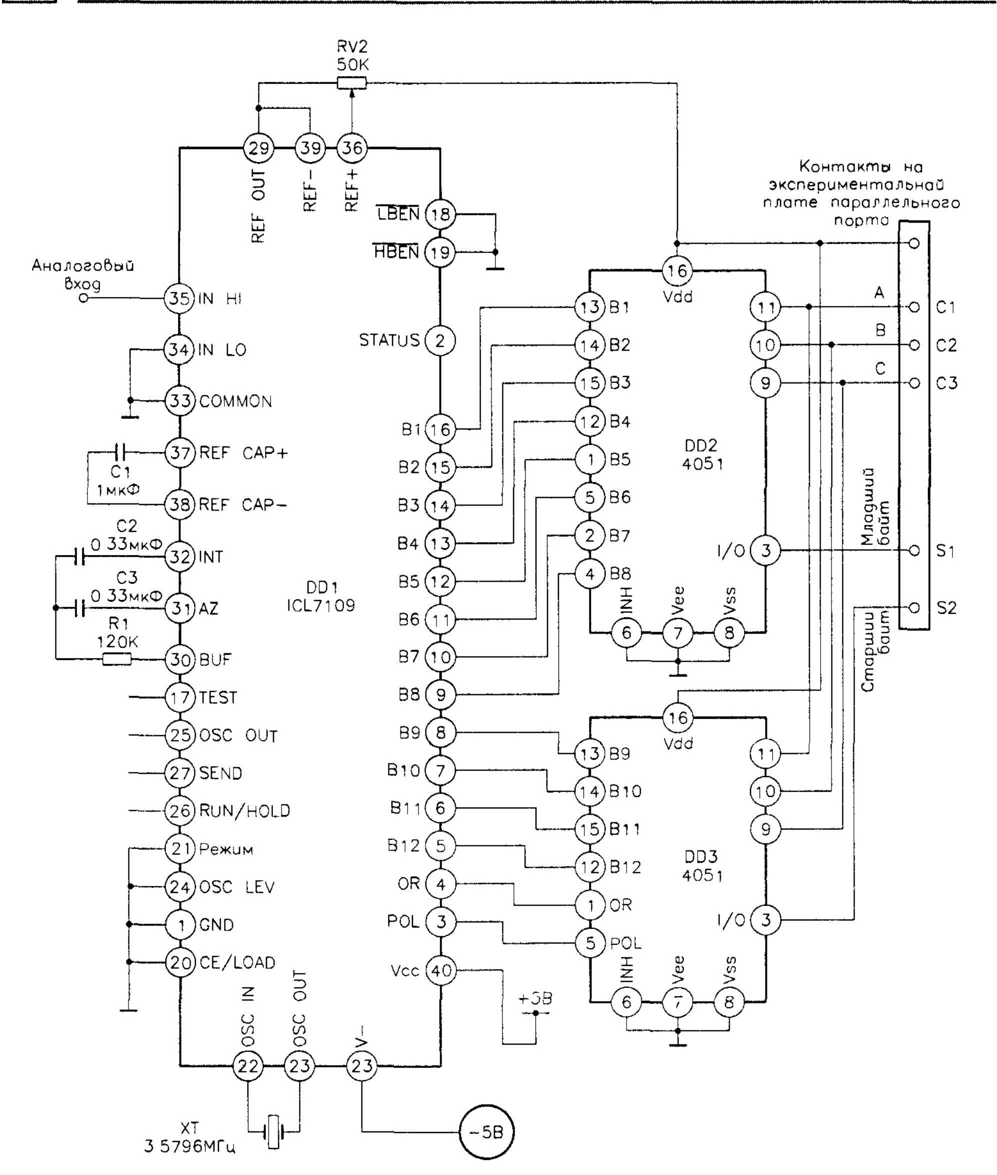


Рис. 6.7. Схема АЦП на микросхеме ICL7109

```
begin
    write_control_port(P_address,i-1),    (*Выбор определенного бита выхода данных 7109 *)
    delay(1),
    low_byte =low_byte+bit_weight(i)*(read_status_port(P_address) and 1)
    high_byte =high_byte+bit_weight(i)*round((read_status_port(P_address) and 2)/2)
end,
if high_byte and 32=32 then polarity =1 else polarity =-1  (*Определение полярности *)
high_byte =high_byte and (1+2+4+8),  (*Содержит четыре старших бита данных АЦП *)
```

```
    read_voltage:=(high_byte*256+low_byte)/2049*2*polarity;
    if high_byte and 16=16 then read_voltage:=999;
end;

(*Главная программа.*)
begin
  centronic_address;
  repeat
      gotoxy(20,10);
      write('Measured input voltage [V] ', read_voltage:6:4);
      delay(9000);
  until keypressed;
end.
```

Интегральные цифровые вольтметры ICL7135

Микросхема ICL7135CPI-2 (Maxim, RS427-483) – прецизионный АЦП с двойным интегрированием и точностью ±1 на 20000 импульсов тактирующего сигнала (рис. 6.8). Эта микросхема идеально подходит для использования в цифровых вольтметрах: напряжение измеряется с точностью до пяти десятичных разрядов. ICL7135 снабжена системой автоматического определения нулевого уровня и полярности, имеет мультиплексированные двоичные выходы. Для построения цифрового вольтметра ее достаточно подключить к светодиодному дисплею. Может также соединяться с компьютером. Напряжение питания ±5 В. Ток потребления от положительной и отрицательной шин равен соответственно 1,1 и 0,8 мА.

Аналоговая секция устройства требует наличия таких внешних элементов, как опорный конденсатор Cref, конденсатор автоматического определения нуля Caz,

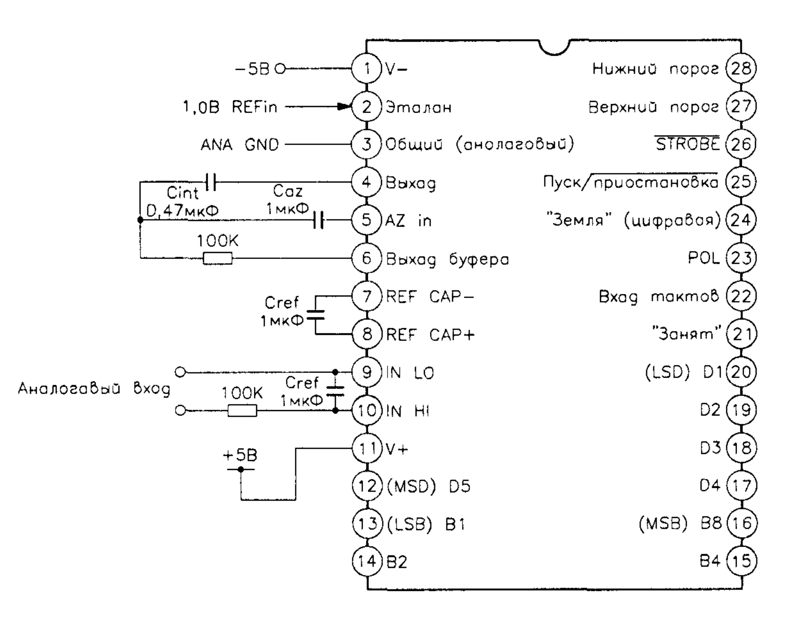


Рис. 6.8. Назначение выводов и типовое включение микросхемы ICL7135

интегрирующий конденсатор Cint и резистор Rint, номиналы которых выбираются в зависимости от конкретного приложения и в соответствии с технической документацией изготовителя.

Последовательность вывода данных приведена на рис. 6.9. Микросхема имеет пять разрядных формирователей (выходы D5 – D1) и шину вывода каждого разряда в двоично-десятичном коде (выходы B8, B4, B2, B1). Вывод измеренного значения напряжения осуществляется поразрядно, начиная со старшего. На выходах разрядных формирователей (от D5) последовательно формируется положительный импульс, его значение синхронно поступает на шину вывода разряда. Выход $\overline{\text{STROBE}}$ (контакт 26) может использоваться для последовательной передачи измеренного напряжения на внешние устройства.

Когда на входе RUN/$\overline{\text{HOLD}}$ (контакт 25) удерживается высокий уровень, АЦП работает с интервалами в 40002 такта между циклами измерений. Если на этот вход подать низкий уровень, то преобразователь завершит текущий цикл измерения и остановится до тех пор, пока на нем удерживается низкий уровень. Кратковременный положительный импульс (менее 300 нс) инициирует новый измерительный цикл.

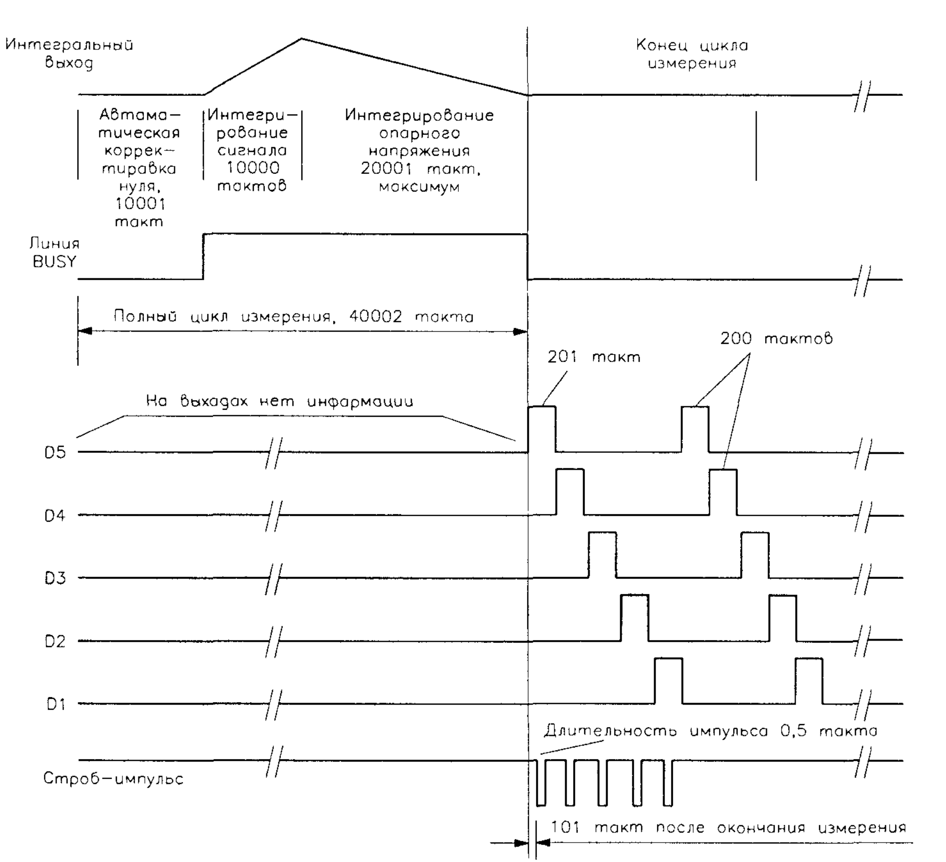


Рис. 6.9. Временные диограммы АЦП ICL7135

Выход BUSY (контакт 21) переходит в единичное состояние с началом интегрирования сигнала и остается в нем, пока напряжение на конденсаторе интегратора не станет равным нулю. Этот выход может использоваться для передачи информации об измеренном напряжении по одному проводу: для вычисления напряжения необходимо измерить количество тактовых импульсов за время, в течение которого BUSY = 1. Затем из полученного значения нужно вычесть 10001 для получения количества тактовых импульсов Ndeint, в течение которых интегрировалось опорное напряжение. После этого входное напряжение можно вычислить по следующей формуле:

$$Vin = \frac{1000}{Ndeint} Vref.$$

Работу микросхемы иллюстрируют две программы. Для функционирования первой из них (она предназначена для подсчета количества тактовых импульсов в течение времени, когда BUSY = 1) ICL7135 подключается к ПК посредством двух линий, также используется экспериментальная плата параллельного порта, где вход BUSY (контакт 21) и тактовый вход CLOCK IN (контакт 22) соединены с контактами S1 и S2. АЦП настроен на постоянную работу, тактовая частота не превышает 50 Гц.

Текст программы ICL7135S.PAS

```
Program ICL7135_single_wire;
uses
  dos;crt;

{$I c:\ioexp\tplib1.pas}

function count_clock:integer;
var
  clock:integer;
  clock_status_old,value:byte;
begin
  clock:=0;
  clock_status_old:=16;
  repeat until port(P_address+1) and 8=0;
  repeat until port(P_address+1) and 8=8;
  repeat
     value:=port(p_address+1) and 16;
     if (clock_status_old=16) and (value=0) then clock:=clock+1;
     clock_status_old:=value;
  until port(P_address+1) and 8=0;
  count_clock:=clock-10001;
end;

(*Главная программа.*)
begin
  centronic_address;
  repeat
     writeln(count_clock);
     delay(1000);
  until keypressed;
end.
```

Вторая программа показывает, как микросхема ICL7135 соединяется с компьютером через UART6402 (рис. 6.10). Выходы B1 – B4, POL, OVER, UNDER и D5 подключены к входам буферного регистра передатчика (TBR0 – TBR7) UART6402. Вход RUN/$\overline{\text{HOLD}}$ соединен с выходом RBR0 буферного регистра приемника. Во время работы последовательные данные передаются из компьютера в UART6402, что переводит RBR0 из нуля в единицу и затем снова в нуль. По положительному фронту этого сигнала микросхема ICL7135 начинает цикл преобразования, после чего пять стробирующих импульсов с выхода $\overline{\text{STROBE}}$ инициируют передачу данных UART. Компьютер за пять раз считывает пять разрядов преобразованных данных. Тактовая частота АЦП равна 153,6 кГц, она вырабатывается кварцевым генератором CD4060 (см. главу 2). UART должен передать данные за 1,3 мс. Если UART работает на скорости 9600 бод, с 8 битами данных, одним стоповым битом и без проверки на четность, то время передачи будет порядка 1 мс. Следовательно, такая настройка UART корректна.

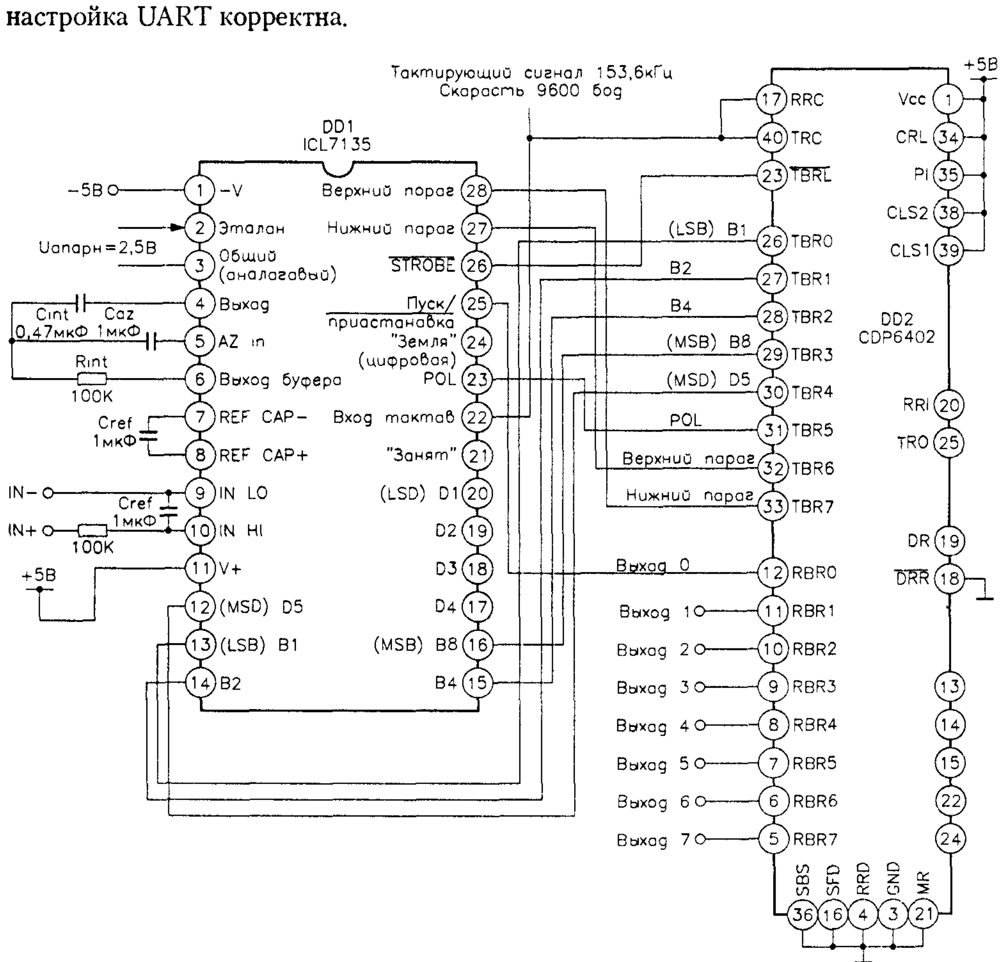


Рис. 6.10. Схемо соединения АЦП ICL7135 с компьютером через UART6402

Текст программы 7135 на VB3

```
'Функции объявляются в DLL, WLIB1.DLL.
'Объявляемые функции:  RS232(), Bit_weight(), Write_interrupt_enable(),
'                      Read_interrupt_identification(),
'                      write_data_format(),     write_transmit_buffer(),
'                      write_modem_status(),    write_receive_buffer(),
'                      read_modem_status().
Declare Function RS232 Lib 'c:\ioexp\wlib.dll' (ByVal X As Integer) As Integer
Declare Function Bit_weight Lib 'c:\ioexp\wlib.dll' (ByVal X As Integer) As Integer
Declare Function Write_interrupt_enable Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer,
ByVal datax As Integer) As Integer
Declare Function Read_interrupt_identification Lib 'c:\ioexp\wlib.dll' (ByVal address As
Integer) As Integer
Declare Function Write_data_format Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer, ByVal
baud As Integer, ByVal parity As Integer, ByVal Data_bit As Integer, ByVal Stop_bit As
Integer) As Integer
Declare Function Write_transmit_buffer Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer,
ByVal datax As Integer) As Integer
Declare Function Write_modem_status Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer, ByVal
RTS As Integer, ByVal DTR As Integer) As Integer
Declare Function Read_receive_buffer Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer) As
Integer
Declare Function Read_modem_status Lib 'c:\ioexp\wlib.dll' (ByVal address As Integer, ByVal X
As Integer) As Integer

Sub Command1_click()
DoEvents
dummy=Read_receive_buffer(RS232_address)
timedelay
dummy=Write_transmit_buffer(RS232_address,128)
timedelay
dummy=Write_transmit_buffer(RS232_address,0)

For i=1 To 5
Do While (Read_interrupt_identification(RS232_address) and 1)=1
Loop
digit(6-i)=Read_receive_buffer(RS232)_address)
Next i

count_voltage=0
For i=1 To 5
count_voltage=count_voltage+10^(i-1)*(digit(i) and 15)
Next i
Label3.Caption=count_voltage
End Sub

Sub Command1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Label7.Caption='Read data from 7135-UART'
End Sub

Sub Command3_Click()
'Настройка выбранного последовательного порта.
baud_rate=Val(text2(0).Text)        'Настройка скорости.
parity=Val(text2(1).Text)           'Настройка проверки на четность.
data_bit_length=Val(text2(2).Text)  'Настройка длины блока данных.
```

```
stop_bit_length=Val(text2(3).Text)  'Настройка длины стоповой посылки.
dummy=write_data_format(RS232_address,baud_rate,parity,data_bit_length,stop_bit_length)
'Запись конфигурации в регистр формата данных.
End Sub

Sub Command3_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Label7.Caption="Change the configuration of RS232 port"
End Sub

Sub Command4_Click()
  End
End Sub

Sub Command4_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Label7.Caption="Quit the program"
End Sub

Sub Command5_Click()
'Выбор другого порта.
dummy=MsgBox(Str(RS232(0)) & "RS232 ports (COMs) are installed on your PC. Their base
addresses are " & Format$(RS232(1),"###") & "    "& Format$(RS232(2),"###") & "    "&
Format$(RS232(3),"###") & "    "& Format$(RS232(4),"###") & "Decimal",48,"RS232 ports (COM) on
your computer")                               'Отображение информации о последовательном порте.
RS232_number=Val(InputBox$("Input 1,2,3 or 4 to select RS232 port (COM) for the Mini-Lab Data
Logger/Controller", "Select COM ports"))      'Выбор последовательного порта.
RS232_address=RS232(RS232_number)             'Получение адреса выбранного порта.
Label2.Caption="Selected COM port:" &Format(RS232_number)    'Отображение информации
                                                             'о выбранном порте.
Label4.Caption="Base address of COM:" & Format(RS232_address)
End Sub

Sub Command5_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  Label7.Caption="Change RS232 port number"
End Sub

Sub Form_Load()
  For i=0 To 11
     status(i)=0
Next i

dummy=MsgBox(Str(RS232(0)) & "RS232 ports (COMs) are installed on your PC. Their base
addresses are " & Format$(RS232(1),"###") & "    "& Format$(RS232(2),"###") & "    "&
Format$(RS232(3),"###") & "    "& Format$(RS232(4),"###") & "Decimal",48,"RS232 ports (COM) on
your computer")
RS232_number=Val(InputBox$("Input 1,2,3 or 4 to select RS232 port (COM) for the Mini-Lab Data
Logger/Controller", "Select COM ports"))

RS232_address=RS232(RS232_number)
Label2.Caption="Selected COM port:" &Format(RS232_number)
Label4.Caption="Base address of COM:" & Format(RS232_address)
baud_rate=96
parity=0
data_bit_length=8
stop_bit_length=1

text2(0).Text=Format$(baud_rate)
text2(1).Text=Format$(parity)
```

```
text2(2).Text=Format$(data_bit_length)
text2(3) Text=Format$(stop_bit_length)

dummy=write_data_format(RS232_address,baud_rate,parity,data_bit_length,stop_bit_length)
dummy=Write_interrupt_enable(RS232_address,1)    'Настройка прерывания.
End Sub

Sub Label3 MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
  'Отображение принятых последовательных данных.
  Label7.Caption="Value of the serial input data"
End Sub

Sub Label6.MouseMove(index As Integer, Button As Integer, Shift As Integer, X As Single, Y As
Single)
Select Case index
Case 0
Label7 Caption= Baud rate=115200,19200,9600,2400 etc."
Case 1
Label7.Caption="0=no parity, 1=Odd parity, 3=Even parity"
Case 2
Label7.Caption="Input 5,6,7 or 8 to select the data bit length"
Case 3
Label7 Caption='Input 1 or 2 to select Stop bit"
End Select
End Sub

Sub Text2 Change(index As Integer)
Select Case index
Case 0:
Case 1.
Case 2:
Case 3.
End Select
End Sub

Sub timedelay()
For i=1 To 100
i=i
Next i
End Sub
```

6.1.2. АЦП с последовательным интерфейсом ввода/вывода

В данном разделе приводится описание микросхем АЦП, реализующих последовательный способ вывода оцифрованного напряжения.

Восьмиразрядный АЦП последовательного приближения TLC548

Микросхема TLC548CP/TLC549CP (Texas Instruments, RS200-6757) – это восьмиразрядный АЦП последовательного приближения (рис. 6.11). Он снабжен встроенной схемой выборки и хранения, тактовым генератором на 4 МГц и последовательным интерфейсом ввода/вывода. Микросхема TLC548 имеет частоту дискретизации 45500, а TLC549 – 40000 Гц. Они взаимозаменяемы и аппаратно совместимы с АЦП TLC1540, который имеет точность 10 разрядов.

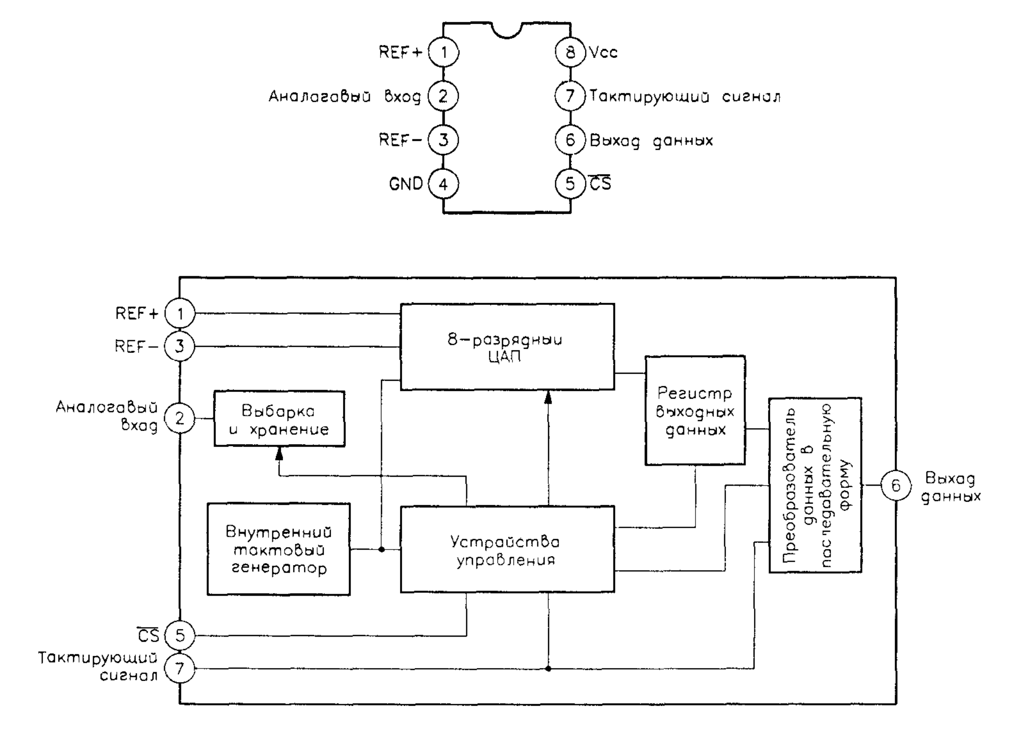


Рис. 6.11. Назначение выводов и внутренняя блок-схема микросхемы TLC548

Входы Vcc (контакт 8) и GND (контакт 4) соединены с положительным и отрицательным проводами источника питания. Напряжение питания может изменяться от 3 до 6 В, потребляемый ток равен 1,9 мА. На входы REF+ и REF− (контакты 1 и 3) подается опорное напряжение. Входы REF− и GND часто соединяются.

Последовательный интерфейс состоит из двух ТТЛ совместимых входных линий, входа тактового сигнала (контакт 7), входа выбора микросхемы ($\overline{CS}$, контакт 5) и линии вывода данных (контакт 6), которая имеет три состояния (рис. 6.12).

Если на вход выбора микросхемы $\overline{CS}$ подан сигнал высокого уровня, то линия вывода данных имеет высокое сопротивление и тактовый вход закрывается. При подаче на вход $\overline{CS}$ низкого уровня начинается цикл преобразования, а старший разряд предыдущего преобразования (DB7) автоматически появляется на выходе (контакт 6).

Отрицательные фронты первых четырех тактовых импульсов последовательно загружают разряды (DB6, DB5, DB4 и DB3) предыдущего преобразования на выход данных. Встроенная схема выборки и хранения начинает дискретизацию после четырех отрицательных фронтов тактовых импульсов.

При прохождении еще трех отрицательных фронтов тактовых импульсов на выходе микросхемы последовательно появляются оставшиеся разряды (DB2, DB1 и DB0) предыдущего преобразования.

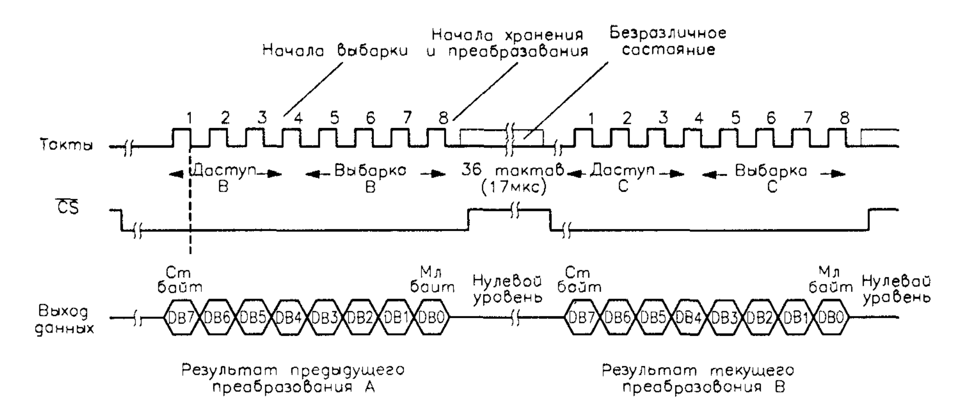


Рис. 6.12. Временные диаграммы работы АЦП TLC548/549

Отрицательный фронт последнего (восьмого) тактового импульса завершает процесс выборки и инициирует процесс хранения, который длится в течение четырех внутренних тактовых импульсов, после чего за последующие 32 тактовых импульса происходит преобразование. Полный процесс трансформации занимает 36 внутренних тактовых импульсов. Во время этого процесса вход $\overline{CS}$ должен иметь высокий уровень или тактовый вход должен оставаться в нулевом состоянии по крайней мере на протяжении 36 внутренних тактов. В принципе во время операций преобразования допускается удерживание входа $\overline{CS}$ на низком уровне, но в этом случае необходимо предпринять специальные меры для предотвращения шума на тактовом входе, поскольку шум может вызвать потерю синхронизации между устройством и внешней схемой сопряжения. Если на вход $\overline{CS}$ подан высокий уровень, его следует удерживать до конца преобразования. Один отрицательный фронт по этому входу способен сбросить микросхему и прервать незавершенный цикл преобразования.

Схема, иллюстрирующая работу АЦП TLC548 совместно с экспериментальной платой параллельного порта, приведена на рис. 6.13. Опорное напряжение 2,5 В подается на АЦП от источника опорного напряжения TLE2425. Тактирующий вход и вход $\overline{CS}$ соединены с контактами C1 и C2 на экспериментальной плате. Преобразованные данные подаются на контакт S1.

Текст программы TLC548.PAS на TP6

```
Program TLC548,
(*Программа управления АЦП с последовательным интерфейсом  АЦП соединен с экспериментальной
платой параллельного порта *)
(*Тактовый вход соединен с C1, CS - с C2, выход данных - с S1 *)
uses
  dos,crt,
var
  1 integer,
{$I c \ioexp\tplib1 pas}
```

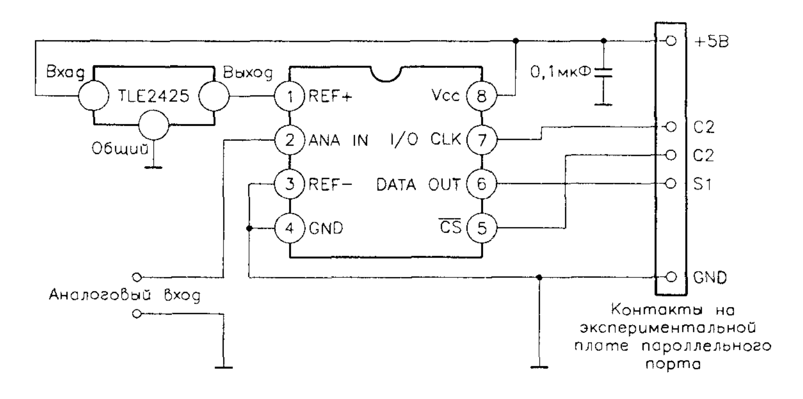


Рис. 6.13. АЦП TLC548

```
function AD_converter real;
(*Процедура аналого-цифрового преобразования.*)
var
  data:array[1..8] of byte;
  ii,ij,addx·byte;
begin
  (*Два считывания для получения имеющегося значения результата преобразования.*)
  for ij:=1 to 2 do
      begin
          (*CS=0 для начала преобразования.*)
          write_data_port(P_address,0+2);   (*Clock=0, CS=1 *)
          write_data_port(P_address,0+0);   (*Clock=0, CS=0.*)
          for ii·=1 to 8 do
              begin                         (*Считывание предыдущего результата *)
                  data[ii]:=read_status_port(P_address) and 1;  (*Считывание цифры.*)
                  write_data_port(P_address,0+0);               (*Clock=0, CS=0 *)
                  write_data_port(P_address,1+0);               (*Clock=1, CS=0.*)
                  write_data_port(P_address,0+0);               (*Clock=0, CS=0.*)
              end;
      end,
(*Вычисление значения входного напряжения. Опорное напряжение равно 2,5 В.*)
AD_Converter·=(128*data[1]+64*data[2]+32*data[3]+16*data[4]+8*data[5]+4*data[6]+2*data[7]+1*data[8])*
2.5/256,
End;

(*Главная программа *)
begin
  centronic_address;
  repeat
      gotoxy(25,10); write('Input voltage [V]:', AD_Converter:6:3);
      delay(2000);
  until keypressed;
end.
```

12-канальный АЦП TLC541

Микросхемы TLC541IN и TLC540IN (Texas Instruments, RS649-289) – это восьмиразрядные АЦП последовательного приближения. Они имеют встроенную схему выборки и хранения, 12-канальный аналоговый мультиплексор и последовательный интерфейс ввода/вывода, позволяющий одновременно производить операции чтения и записи.

Микросхема TLC540 (рис. 6.14) характеризуется частотой дискретизации 75180 Гц, а TLC541 – 40000 Гц. Эти АЦП можно заменить на TLC1540 и TLC1541, которые имеют точность 10 разрядов и полностью с ними совместимы; их частота дискретизации составляет 32258 Гц. Выводы Vcc (контакт 20) и GND (контакт 10) соединены с положительным и отрицательным проводами источника питания. Напряжение источника питания может изменяться от 4,75 до 6,5 В, потребляемая мощность равна 6 мВт. На входы REF+ и REF– (контакты 14 и 13) подается внешнее опорное напряжение. Входы REF– и GND, как правило, соединены.

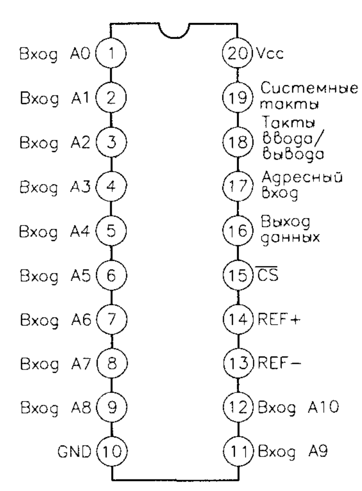


Рис. 6.14. Назначение выводов микросхемы TLC541

К первым 11 аналоговым мультиплексорам можно получить доступ через контакты 1–9, 11 и 12, соответствующие аналоговым входам 0–11. Двенадцатый вход соединен с опорным напряжением для реализации режима самотестирования. Чтобы выбрать один из 12 входов, в микросхему необходимо записать четырехразрядный адрес через последовательный интерфейс.

Последовательный интерфейс состоит из пяти ТТЛ совместимых линий ввода/вывода: входа системных тактов (SYSTEM CLOCK, контакт 19), входа тактов ввода/вывода (I/O CLOCK, контакт 18), входа выбора микросхемы ($\overline{CS}$, контакт 15), адресного входа (ADDRESS INPUT, контакт 17) и выхода данных (DATA OUT, контакт 16). SYSTEM CLOCK – это вход тактовых импульсов для операций преобразования. В АЦП TLC540 их максимальная частота равна 4 МГц, в TLC541 – 2,1 МГц, частота дискретизации – 75180 и 40000 Гц соответственно. У микросхем TLC1540 и TLC1541 максимальная частота тактовых импульсов составляет 2,1 МГц, частота дискретизации – 32258 Гц. Вход I/O CLOCK используется при синхронизации операций ввода/вывода, ADDRESS INPUT – при выборе аналогового канала. DATA OUT – это последовательный выход оцифрованного аналогового напряжения, $\overline{CS}$ – вход выбора микросхемы. Для разрешения работы микросхемы на этот вход требуется подать сигнал низкого уровня. Если $\overline{CS}$ = 1, то выход DATA OUT находится в третьем состоянии, а входы ADDRESS INPUT и I/O CLOCK закрыты, что позволяет подключать несколько подобных устройств к одной общей шине.

Входы SYSTEM CLK и I/O CLOCK используются независимо. Временные диаграммы чтения и записи представлены на рис. 6.15.

На вход $\overline{CS}$ подается сигнал низкого уровня. Старший разряд предыдущего результата преобразования (DB7) автоматически появляется на выходе DATA OUT.

Новый адрес аналогового канала (AD0, AD1, AD2 и AD3) загружается в микросхему при прохождении первых четырех положительных фронтов импульсов на входе I/O CLOCK. Старший разряд адреса (AD3) загружается первым. Отрицательные фронты сигнала I/O CLOCK сдвигают разряды DB6, DB5, DB4 и DB3 предыдущего результата преобразования на выход. После четвертого отрицательного фронта сигнала I/O CLOCK начинается преобразование сигнала с аналогового входа, заданного новым адресом.

На вход I/O CLOCK подаются три тактовых импульса. При этом разряды DB2, DB1 и DB0 предыдущего результата преобразования сдвигаются по прохождении каждого отрицательного фронта на выход.

На вход I/O CLOCK подается последний (восьмой) тактовый импульс, отрицательный фронт которого завершает процесс выборки и инициирует процесс хранения. Процесс преобразования занимает 36 тактов на входе SYS CLOCK. Затем для разрешения нового преобразования на вход $\overline{CS}$ подается высокий уровень либо на вход I/O CLOCK – низкий, по крайней мере, в течение 36 тактовых интервалов. На входе $\overline{CS}$ допустимо удерживать сигнал низкого уровня, но следует принять специальные меры для предотвращения шума на входе I/O CLOCK, так как это может привести к потере синхронизации между устройством и внешней схемой сопряжения. Если на входе $\overline{CS}$ сигнал высокого уровня, он должен удерживаться в таком состоянии до конца преобразования. Случайный отрицательный фронт по этому входу способен сбросить микросхему и прервать незавершенный цикл преобразования.

Схема на рис. 6.16 показывает, как описанный АЦП соединяется с экспериментальной платой параллельного порта. Опорное напряжение 2,5 В генерируется источником опорного напряжения TLE2425. Синхросигнал может быть сформирован

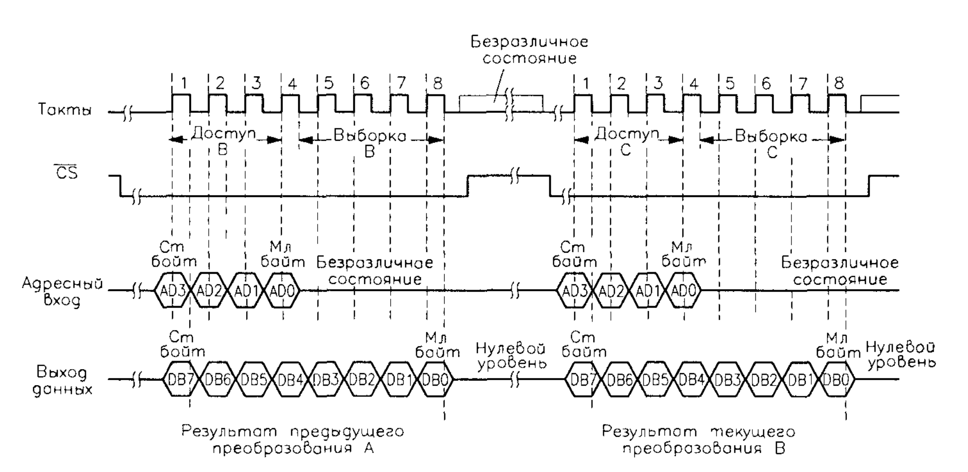


Рис. 6.15. Временные диаграммы работы АЦП TLC541

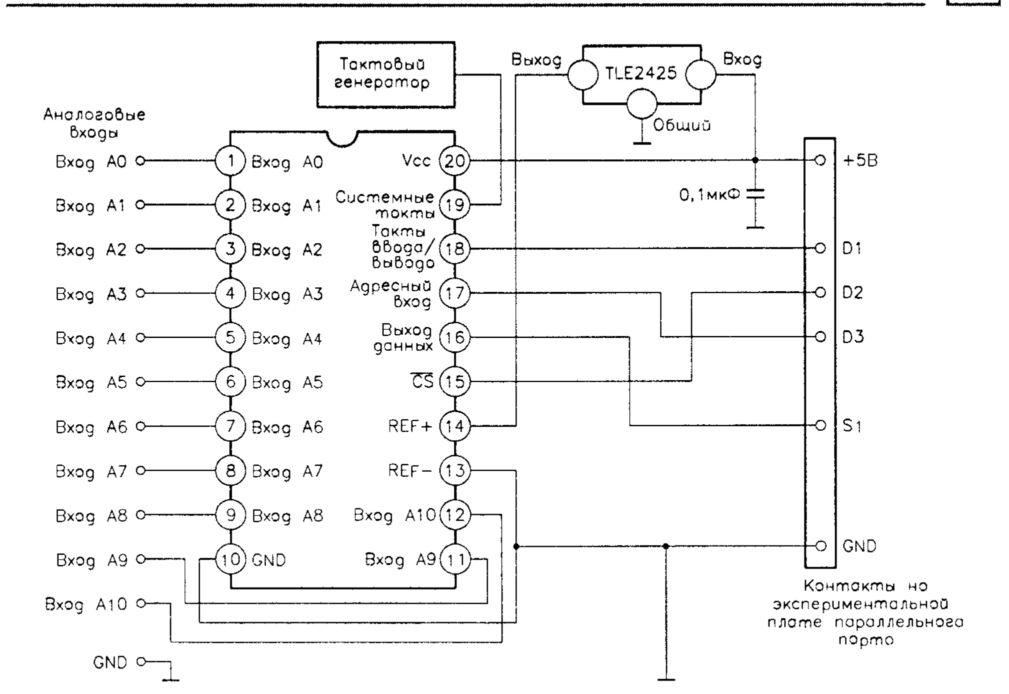


Рис. 6.16. Экспериментальная схема на базе АЦП TLC541

схемой на базе интегрального таймера 555 (см. главу 2). Входы I/O CLOCK, $\overline{CS}$ и DATA OUT TLC541 соединены с контактами D1, D2 и D3 на экспериментальной плате, выход данных – с контактом S1 на плате.

Текст программы TLC541.PAS на TP6

```
Program TLC541,
(*Программа управления АЦП с последовательным интерфейсом TLC541. АЦП соединен с экспериментальной
платой параллельного порта.*)
(*Вход I/O CLOCK соединен с контактом D1, CS - с D2, адресный вход - с D3, выход данных - с S1.*)
uses
dos,crt;

var
i:integer;
dummy real;
{$I c:\ioexp\tplib1.pas}

Function AD_converter(address:byte):real;
(*Процедура аналого-цифрового преобразования. Адреса: 0-11.*)
var
   add,data:array[1..8] of byte;
   ii,addx.byte;
begin
   (*Нахождение битов адреса.*)
   for ii:=1 to 8 do add[ii]:-0;
   if address>=8 then begin add[1]:=1; address:=address-8; end;
```

```
    if address>=4 then begin add[2] =1, address =address-4  end,
    if address>=2 then begin add[3] =1, address =address-2, end,
    if address>=1 then begin add[4] =1, end,
    (*CS, бит 2, переводится в нулевое состояние для начала преобразования *)
    write_data_port(P_address 4),
    write_data_port(P_address,0),
    delay(1),

    for ii =1 to 8 do   begin                        (*Чтение предыдущего результата *)
                            addx =add[ii]*2,
                            data(ii] =read_status_port(P_address) and 1,  (*Чтение цифры *)
                            write_data_port(P_address,addx),       (*Загрузка бита адреса
                                                                    I/O CLK=0 *)
                            write_data_port(P_address,1+addx),     (*I/O CLK=1 *)
                            write_data_port(P_address,addx),       (*I/O CLK=0 *)
                            delay(1),
                        end
    AD_converter =(128*data[1]+64*data[2]+32*data[3]+16*data[4]+8*data[5]+4*data[6]+2*data[7]+
    1*data[8])*2 5/255,
  end,

  (*Главная программа *)
  begin
    Centronic_address,
    Repeat
       for i =0 to 11 do
       begin
          dummy =AD_Converter(i),                              (*Запуск АЦП в первый раз *)
          delay(1),
          gotoxy(20,5+i), write( Input voltage [V] to Channel [ ,i 2, ]  ,AD_converter(i) 6 3),
          (*АЦП считывает результаты предыдущего преобразования *)
       end,
    delay(2000),
    until keypressed,
  end
```

12-разрядный АЦП последовательного приближения LTC1288

Микросхема LTC1288CN8 (Linear Technology, RS197-1795) – это маломощный 12-разрядный аналого-цифровой преобразователь последовательного приближения (рис. 6.17). Напряжение источника питания (2,7–6 В) подается на контакты 8 (плюс) и 4 (минус). Контакт 8 также служит входом опорного напряжения, поэтому источник питания по шумам и пульсациям должен быть высокого качества. Ток потребления равен 260 мкА при частоте дискретизации 6,6 кГц и напряжении питания 2,7 В. Когда микросхема находится в режиме ожидания, ток потребления снижается до нескольких наноампер. В этом АЦП имеется два аналоговых входа (контакты 2 и 3), которые могут быть настроены на два режима: в первом входное напряжение подается на каждый вход относительно «земли» (режим несимметричного ввода), во втором – на два входа (дифференциальный режим) Ток потребления аналогового входа составляет 1 мкА.

Микросхема LTC1288 соединяется с другими внешними схемами с помощью четырехпроводного последовательного интерфейса. Вывод $\overline{CS}$/SHDN (контакт 1) – это вход выбора микросхемы (низким уровнем). Когда на нем высокий уровень,

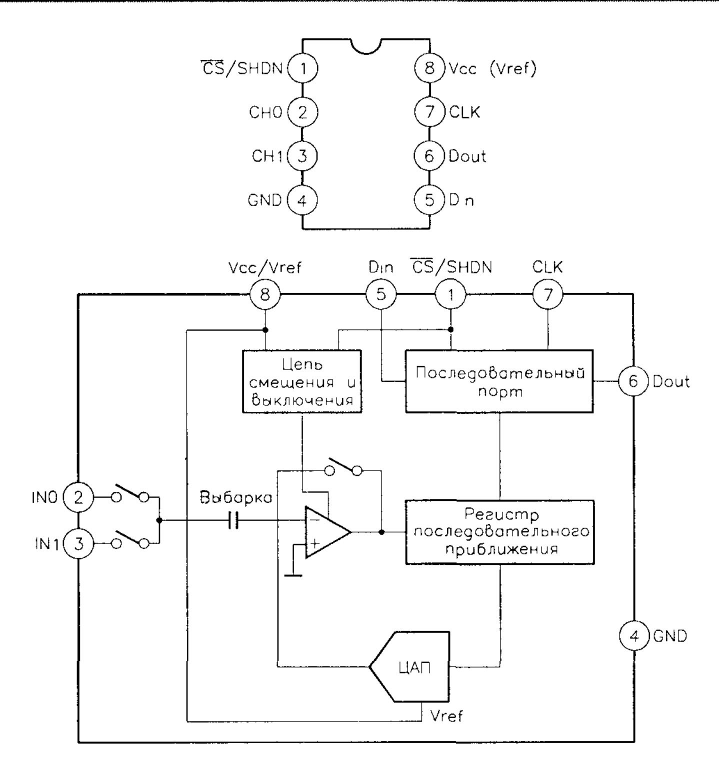


Рис. 6.17. Назначение выводов и внутренняя блок-схема микросхемы LTC1288CN8

микросхема находится в режиме ожидания. На вход CLK (контакт 7) подается тактовый сигнал, синхронизирующий последовательную передачу данных и определяющий скорость преобразования. Каждый разряд результата преобразования передается во время прохождения отрицательного фронта импульса CLK. При прохождении положительного фронта блокируется ввод данных в микросхему. Din (контакт 5) – цифровой вход, который используется для последовательной загрузки адреса выбранного аналогового входа. Dout (контакт 6) – выход цифровых данных, на котором появляется результат преобразования.

Временные диаграммы работы АЦП LTC1288 приведены на рис. 6.18. Цикл начинается при прохождении отрицательного фронта импульса по входу выбора микросхемы $\overline{CS}$/SHDN. Затем микросхема ждет стартовый бит – положительный импульс на входе Din (контакт 5), который загружается в LTC1288 по положительному фронту тактового импульса CLK. Далее в микросхему поступает трехразрядное слово (биты 1–3) для настройки режима ввода и формата вывода последовательных данных. Преобразование начинается при прохождении отрицательного фронта четвертого такта. Сразу после этого на выходе Dout (контакт 6) появляется сигнал низкого уровня. При следующих 12 отрицательных фронтах тактовых импульсов на выходе Dout появляются 12 разрядов результата преобразования. Изменение сигнала на входе Din не влияет на результат.

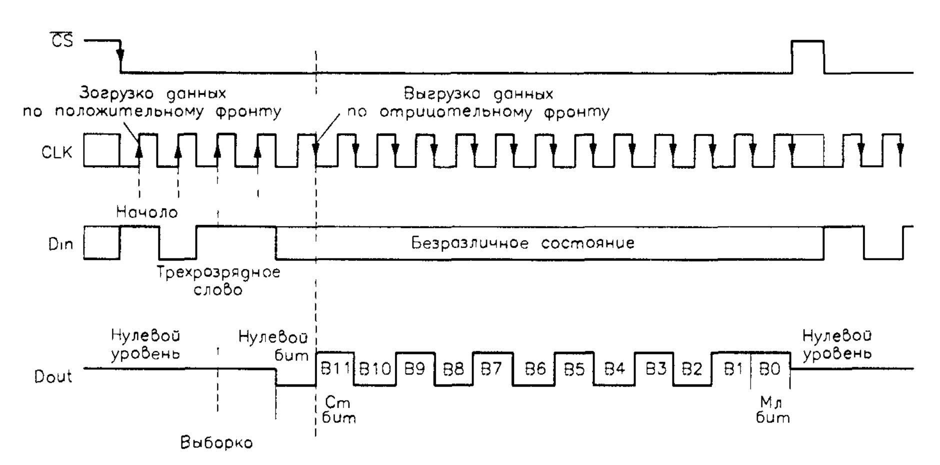


Рис. 6.18. Временные диаграммы работы АЦП LTC1288

Функции битов трехразрядного входного слова приведены ниже. Биты 1 и 2 настраивают аналоговый режим ввода. Бит 3 предназначен для выбора формата выходных данных.

бит 1 = 1, бит 2 = 0	измерение напряжения между каналом 0 и «землей» (несимметричный ввод)
бит 1 = 1, бит 2 = 1	измерение напряжения между каналом 1 и «землей» (несимметричный ввод)
бит 1 = 0, бит 2 = 0	измерение напряжения между каналами 0 и 1 (дифференциальный ввод)
бит 1 = 0, бит 2 = 1	измерение напряжения между каналами 1 и 0 (дифференциальный ввод)
бит 3 = 1	сдвиг битов результата преобразования от старшего к младшему (B11 – B0)
бит 3 = 0	сдвиг битов результата преобразования от младшего к старшему (B0 – B11)

Пример использования микросхемы совместно с экспериментальной платой параллельного порта показан на рис. 6.19. Выводы CLOCK, $\overline{CS}$, DATA IN и DATA OUT соединены с контактами D1, D3, D2 и S1. Текст программы управления написан на языке TP6.

Текст программы LTC1288.PAS

```
Program LTC1288,
(*Программа управления АЦП с последовательным интерфейсом LTC1288  АЦП соединен
с экспериментальной платой параллельного порта *)
(*Вывод CLOCK соединен с D1, CS - с D3, DATA IN - с D2, DATA OUT - с S1 *)
uses
  dos,crt,

{$I c \ioexp\tplib1 pas}
```

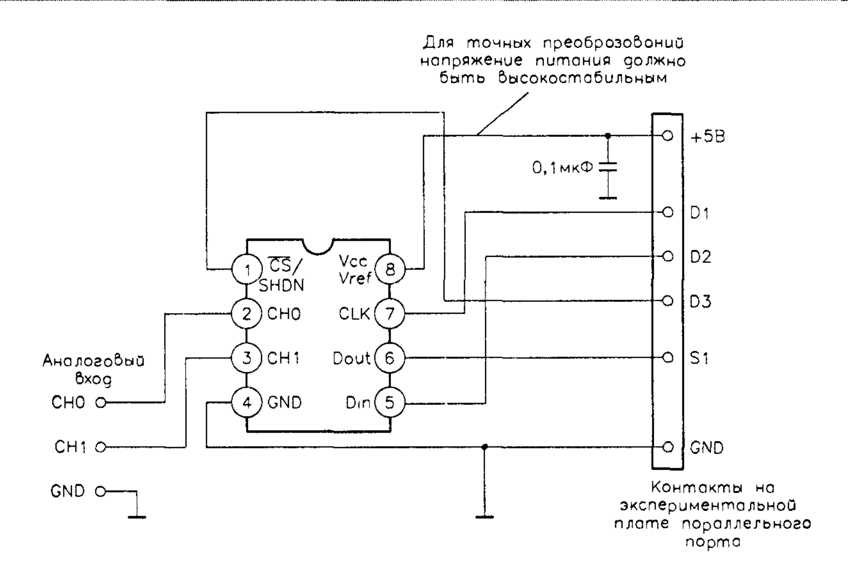


Рис. 6.19. Схема с использованием микросхемы LTC1288

```
function AD_converter(mode·integer):integer,
(* Режим 1, режим несимметричного ввода, канал 0.
   Режим 2, режим несимметричного ввода, канал 1.
   Режим 3, дифференциальный режим, канал 0 положительный, канал 1 отрицательный.
   Режим 4, дифференциальный режим, канал 1 положительный, канал 0 отрицательный.*)

var
   ii,single_differential,odd_sign,dummy_byte·byte;
   IO_data·array[1. 12] of byte;
   Data·array[1. 12] of integer;
   Digital_data:array[1..12] of byte;
   Binary_weight,dummy·integer;

Procedure delay,
{Кратковременная задержка.}
var
   ij:integer;
begin
   for ij:=1 to 6 do ij:=ij;
end;

procedure AD_control(datax:byte);
{Выдача бита для управления АЦП }
begin
   write_data_port(P_address,0+2*datax),      {CLK=0, Dout=datax, стартовый бит=1.}
   write_data_port(P_address,1+2*datax);      {CLK=1, Dout=datax, стартовый бит
                                              заносится в АЦП }
   write_data_port(P_address,0+2*datax),      {CLK=0, Dout=datax }
   delay;
end,

procedure configure_mode,
{Присвоение значений переменным ODD_sign и Single_differential }
```

```
begin
  case mode of
  1: begin odd_sign:=0; Single_differential:=1; end;
  2: begin odd_sign:=1; Single_differential:=1; end;
  3: begin odd_sign:=0; Single_differential:=0; end;
  4: begin odd_sign:=1; Single_differential:=0; end;
  else begin odd_sign:=0; Single_differential:=1; end;
end;
end;

begin
  configure_mode;                     (*Определение битов и Single_differential.*)
  binary_weight:=4096;
  write_data_port(P_address,1+2+4);   (*CLOCK=1, Data=1, CS=1.*)
  delay;
  write_data_port(P_address,1+2+0);   (*CLOCK=1, Data=1, CS=0. Начало цикла преобразования.*)

  (*Ввод бита управления в АЦП.*)
  AD_control(1);                      (*Ввод стартового бита.*)
  AD_control(Single_differential);    (*Ввод бита Single_differential.*)
  AD_control(ODD_sign);               (*Ввод бита выбора канала.*)
  AD_control(1);                      (*Ввод младшего бита управления.*)

  (*Считывание битов данных от B11 до B0.*)
  for ii:=1 to 12 do
  begin
      binary_weight:=binary_weight div 2;
      write_data_port(P_address,1+2);  (*CLOCK=1, Dout=1, CS=0.*)
      delay;
      write_data_port(P_address,0+2);  (*CLOCK=0, Dout=1, CS=0.*)
      delay;
      data[ii]:=(read_status_port(P_address) and 1)*binary_weight;  (*Считывание бита
                                                                     данных.*)
  end;
(*Все 12 бит результата преобразования переданы.*)
  write_data_port(P_address,1+2+4);   (*После завершения преобразования CLOCK=1,
                                       Data=1, CS=1.*)

  dummy:=0;
  for ii:=1 to 12 do dummy:=dummy+data[ii];
  AD_converter:=dummy;
end;

(*Главная программа.*)
begin
  Centronic_address;
  repeat
      gotoxy(20,10); write('Input voltage [V] to channel [0]:',
                           AD_converter(1)*5.06/4096:6:4);
      gotoxy(20,11); write('Input voltage [V] to channel [1]:',
                           AD_converter(2)*5.06/4096:6:4);
      (*Опорное напряжение = напряжение питания = 5,06 В.*)
      delay(2000);
  until keypressed;
end.
```

6.1.3. Аналоговый процессор АЦП TSC500

Микросхема TSC500ACPE (Telcom, RS656-697) – это законченный 16-разрядный АЦП с двойным интегрированием, который включает все аналоговые схемы, необходимые для оцифровки аналогового напряжения (рис. 6.20).

Точность преобразования можно регулировать. При низкой точности достигаются достаточно высокие скорости преобразования, при высокой – наоборот. Разрешение и скорость преобразования допустимо изменять программно.

Микросхема имеет два цифровых входа: A (контакт 12) и B (контакт 13) – и один цифровой выход COMP OUT (контакт 14). С помощью входов A и B выбирается один из четырех этапов работы микросхемы, которые соответствуют следующим четырем фазам:

- A=0, B=0 – фаза вывода нуля интегратора;
- A=0, B=1 – фаза автоопределения нуля;
- A=1, B=0 – фаза интегрирования аналогового сигнала;
- A=1, B=1 – фаза интегрирования опорного напряжения.

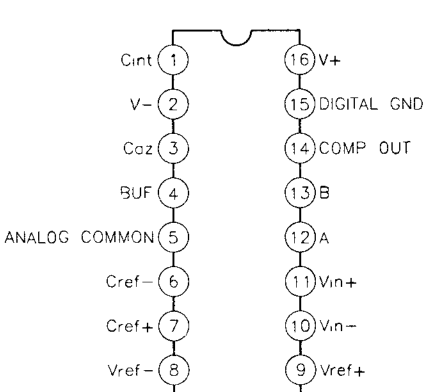


Рис. 6.20. Назначение выводов микросхемы TSC500ACPE

Контакт 14 – это цифровой выход микросхемы, который сигнализирует о завершении операции интегрирования опорного напряжения. Для работы устройства необходимо наличие следующих внешних элементов: опорного и интегрирующего конденсаторов, конденсатора автоопределения нуля, интегрирующего резистора и внешнего источника опорного напряжения (см. рис. 6.22). Аналоговые входы соединены с Vin+ (контакт 11) и Vin– (контакт 10). Для данной микросхемы требуется двуполярный источник питания. При работе от напряжения питания ±5 В входы и выходы совместимы с логикой ТТЛ, ток потребления равен 1 мА.

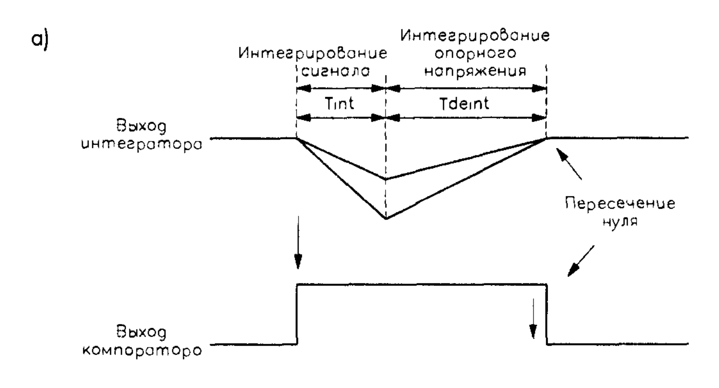
Фазы работы выбираются в следующем порядке:

1. Автоопределение нуля.
2. Интегрирование входного напряжения сигнала.
3. Интегрирование опорного напряжения.
4. Вывод нуля интегратора.

Временные диаграммы работы приведены на рис. 6.21.

Фаза автоопределения нуля используется для компенсации ошибок, связанных с напряжением смещения нуля во внутренней аналоговой секции микросхемы. Затем начинается фаза интегрирования входного аналогового сигнала.

На этом этапе интегрируется напряжение между входами Vin+ и Vin–. Интегрирующий конденсатор заряжается до определенного напряжения. Полярность входного сигнала отображается на выходе COMP OUT (контакт 14). Если



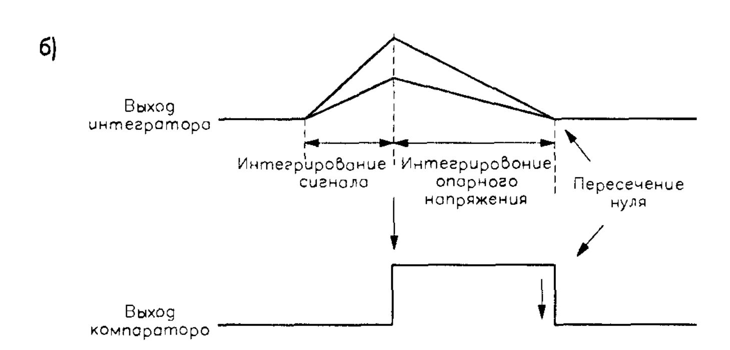


Рис. 6.21. Временные диаграммы работы микросхемы TSC500ACPE. а – входной сигнал положительный, б – входной сигнал отрицательный

входное напряжение положительное, здесь устанавливается высокий уровень сразу после начала фазы интегрирования. Если напряжение отрицательное, на выходе COMP OUT низкий уровень остается до тех пор, пока не начнется этап интегрирования опорного напряжения. Длительность периода интегрирования входного напряжения на рис. 6.21 обозначена Tint. Во время интегрирования опорного напряжения на выходе COMP OUT остается высокий уровень, а напряжение на выходе интегратора снижается до нуля, причем, когда оно переходит через нуль, выход COMP OUT сигнализирует об этом низким уровнем. Длительность периода интегрирования опорного напряжения на рисунке обозначена Tdeint. Величину входного напряжения можно вычислить по формуле:

$$\text{Входное напряжение} = \text{Vref}\ \frac{\text{Tdeint}}{\text{Tint}}.$$

Номиналы внешних элементов следует выбирать в соответствии с рекомендациями изготовителя.

На рис. 6.22 изображена схема на основе ИС TSC500ACPE, соединенной с экспериментальной платой параллельного порта.

Входы A и B соединены с контактами D1 и D2 на экспериментальной плате. Выход COMP OUT подключен к контакту S1. Сначала компьютер инициирует этап автоопределения нуля на 50 мс. Затем начинается этап интегрирования входного сигнала, при этом микросхема TSC500 работает ровно 40 мс. ПК в данном случае использует встроенный таймер/счетчик 8253. Полярность входного сигнала устанавливается посредством опроса выхода COMP OUT с контакта S1. Далее компьютер начинает интегрирование опорного напряжения. Он считает количество тактовых интервалов до момента, когда напряжение на выходе COMP OUT станет равным нулю. Полученный результат, в том числе 40 мс для интегрирования входного сигнала, используется для определения входного напряжения. Номиналы элементов выбраны в соответствии с технической документацией производителя.

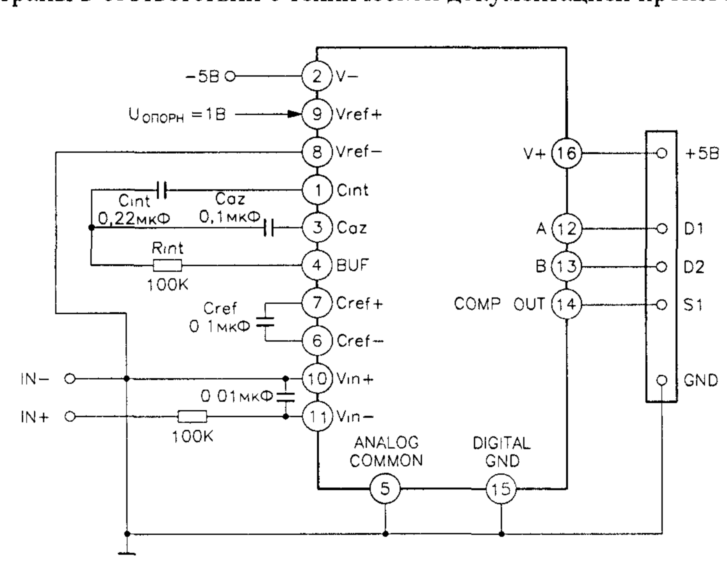


Рис. 6.22. Экспериментальная схема TSC500ACPE

Текст программы TC500A.PAS

```
Program TC500A,
(*Программа управления процессором АЦП с двойным интегрированием
  АЦП соединен с экспериментальной платой параллельного порта,
  A (контакт 12) - с D1, B (контакт 13) - с D2 *)

(*COMP OUT соединен с S1 *)
uses
dos,crt,
var
i,polarity integer;
dummy real,
```

```
{$1 c \ioexp\tplib1 pas }

Procedure init_8253(low_count_byte,high_count_byte byte),
(*Загрузка low_byte и high_byte в третий таймер 8253 *)
(*Тактовая частота 8253  2x1193180 = 2386360 Гц  период тактовых импульсов 1/f = 0 419 нс *)

begin
(*Управляющее слово = b6h = 10110110b
10 = выбор счетчика 2,
11 = чтение/запись сначала младшего байта, а потом старшего,
011 = режим 3,
0 = двоичный счет на 16 *)
  port($43) =$b6,              (*Загрузка управляющего слова в регистр 8253 *)
  port($42) =low_count_byte,   (*Загрузка младшего байта *)
  port($43) =high_count_byte,  (*Загрузка старшего байта *)
  port($61) =port($61) or 1,   (*Отключение внутреннего динамика *)
  port($43) =$80,              (*80H - команда фиксации для третьего счетчика *)
end,

Procedure delay_8253(low_bytex,high_bytex byte),
(*Временная задержка, использующая третий таймер 8253) *)
(*Время задержки определяется переменными low_bytex и high_bytex *)
var
low_byte,high_byte byte,
begin
  init_8253(low_bytex,high_bytex),
  repeat dummy =port($42) until (port($42)=0),
  repeat low_byte =port($42), high_byte port($43), until low_byte<5,
end,

Function deintegration_counts real,
(*Определение времени интегрирования (количества тактов третьего счетчика 8253) *)
var
counts,low_byte,high_byte byte,
finished_flag boolean,
begin
  counts =0,
  repeat
     init_8253(255,255),       (*Загрузка числа 255 в младший
                               и старший регистры счетчика 8253 *)
     repeat
        low_byte =port($42),
        high_byte =port($43),
        if port(P_address+1) and 8=0 then finished_flag =true else finished_flag =false,
     until (low_byte<25) and (hign_byte=0) or finished_flag,
     if not finished_flag then counts =counts+1
  until finished_flag,
  deintegration_counts =counts*(255*256+255) + (255 0-high_byte)*256+(255-low_byte),
end,

Function Voltage real,
(*Определение входного напряжения *)
var
add,data array[1  8] of byte
ii,addx byte,
```

```
begin
  write_data_port(P_address,0+2);  (*Автоопределение нуля в течение 500 мс.*)
  delay(500);
  write_data_port(P_address,1+0);  (*Интегрирование сигнала в течение 40 мс.*)
  delay_8253(117,186);
  polarity:=read_status_port(P_address) and 1;
  delay_8253(117,186);             (*Задержка на 186 × 256 + 117 тактов = 20 мс.*)
  write_data_port(P_address,1+2);  (*Интегрирование опорного сигнала.*)
  dummy:=deintegration_counts;
  if polarity:=0 then polarity:=-1;
  voltage:=polarity*dummy/(186*256+117)/2*1.5;  (*Опорное напряжение - 1,5 В.*)
  delay(100);
  write_data_port(P_address,0);
  delay(50);
end;

(*Главная программа.*)
begin
  Centronic_address;
  write_data_port(P_address,0+2);              (*Автоопределение нуля.*)
  write('Press RETURN to start sampling');readln;
  clrscr;
  repeat
     gotoxy(20,10); write('Input voltage to the TC500:',voltage:6:4);
     delay(1000);
  until keypressed;
end.
```

6.2. Преобразователи напряжение–частота

Преобразователи напряжение–частота – это устройства, которые трансформируют входное напряжение в последовательность прямоугольных импульсов, их частота прямо пропорциональна величине входного напряжения. В данном разделе представлены такие преобразователи и различные способы считывания полученного результата в компьютер.

6.2.1. Принципы преобразования напряжение–частота

Преобразователь напряжение–частота (рис. 6.23) состоит из источника коммутируемого тока, входного компаратора и одновибратора. Компаратор сравнивает положительное входное напряжение V1 с напряжением Vx. Если V1 > Vx, запускается одновибратор, который вырабатывает один импульс фиксированной длительности.

Импульс, формируемый одновибратором, подается через транзисторный ключ на выход схемы и одновременно коммутирует выход генератора тока с входом компаратора на время t = 1,1RtCt. В течение этого времени ток генератора заряжает конденсатор C1, и напряжение Vx становится больше V1. В конце этого временного интервала одновибратор сбрасывается, отключая источник тока от входа компаратора. Затем конденсатор C1 начинает разряжаться через резистор R1. Когда напряжение Vx на конденсаторе C1 сравняется с напряжением V1, компаратор срабатывает, запуская одновибратор, и цикл повторяется. Таким образом, на выходе схемы формируется частота, пропорциональная входному напряжению.

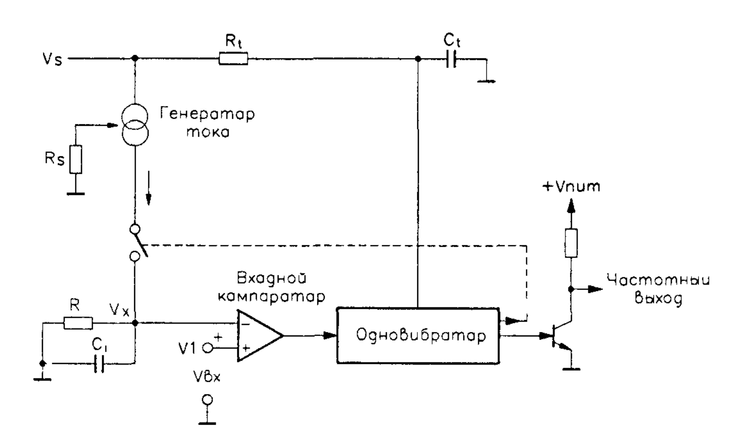


Рис. 6.23. Принцип работы преобразователя напряжение–частота

6.2.2. Преобразователь напряжение–частота LM331

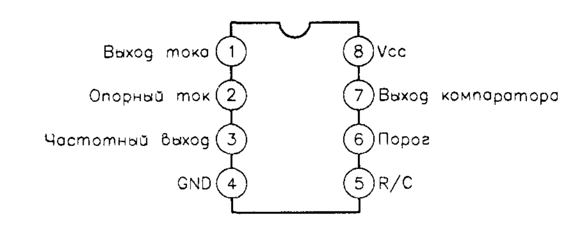
Нелинейность преобразования напряжения в частоту у микросхемы LM331 (National Semiconductor, RS411-652) составляет 0,01% на шкале частот 1 Гц – 100 кГц. Если напряжение питания микросхемы равно +5 В, то ее выход совместим с ТТЛ и КМОП логикой. Выход можно нагружать на три ТТЛ входа. На контакты 4 и 8 подается напряжение питания микросхемы, и они соответственно соединяются с отрицательным и положительным проводами источника питания.

Напряжение источника питания составляет от 4 до 40 В. Ток потребления – 3 мА при напряжении питания 5 В. На контакт 2 внутренней схемой подается опорное напряжение 1,9 В. Резистор установки величины тока Rs включается между выводом 2 и «землей». При этом ток через резистор равен 1,9/Rs. К контакту 5 подключается RC-цепочка, управляющая длительностью импульса одновибратора. На контакт 7 подается входной сигнал, а с помощью контакта 6 устанавливается порог. Контакт 3 – выход микросхемы с открытым коллектором. Выходной ток ограничен порогом 50 мА.

Схема с применением преобразователя LM331, подключенного к экспериментальной плате параллельного порта, изображена на рис. 6.24. Частотный выход соединен с контактом S1. Программа управления измеряет частоту выходного сигнала микросхемы LM331, для этого используется встроенный в компьютер таймер/счетчик 8253.

Текст программы LM331.PAS

```
Program LM331_Voltage_Frequency,
(*Программа управления преобразователем LM331 *)
uses
graph,crt dos,
var
time_period real,
(*Загрузка файла библиотеки *)
{$I c \ioexp\tplib1 pas}
Procedure init_8253(low_count_byte high_count_byte byte)
```



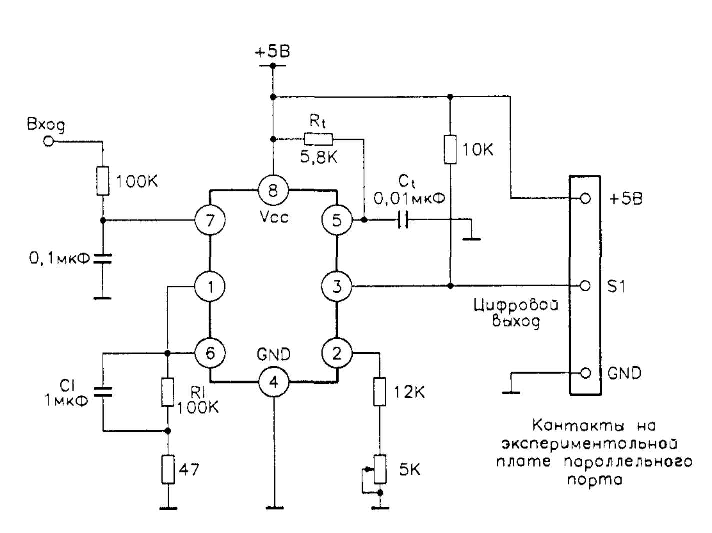


Рис. 6.24. Схема на преобразователе LM331

```
(*Загрузка low_byte и high_byte в третий таймер 8253.*)
{*Тактовая частота 8253: 2×1193180 = 2386360 Гц, период тактовых импульсов 1/f = 0,419 нс.*)
begin
(*Управляющее слово = b6h = 10110110b:
10 = выбор счетчика 2;
11 = чтение/запись сначала младшего байта, а потом старшего;
011 = режим 3;
0 = двоичный счет на 16.*)
   port($43):=$b6;             (*Загрузка управляющего слова в регистр 8253.*)
   port($42):=low_count_byte;  (*Загрузка младшего байта.*)
   port($43):=high_count_byte; (*Загрузка старшего байта.*)
   port($61):=port($61) or 1;  (*Отключение внутреннего динамика.*)
   port($43).=$80;             (*80H - команда фиксации для третьего счетчика.*)
end;
Function read_8253:integer;
(*Считывание двух восьмибитовых регистров.*)
var
low_byte,high_byte:byte;
begin
   low_byte:=port($42);
   high_byte'=port($43);
   read_8253.=low_byte+256*high_byte;
end;
```

```
Function find_period(Address integer,Bit_weight integer) real,
(*Определение периода входного цифрового сигнала
Входной сигнал зависит от адреса входного порта (Address) и бита
Бит 0, Bit_weight=1
Бит 1, Bit_weight=2

Бит 7, Bit_weight=128 *)
var
count Average_number,time1,time2 integer,
begin
(*Определение периода цифрового сигнала  Он будет использоваться для установки значения
переменной Average_number *)
   repeat until port(Address) and Bit_weight=Bit_weight,          (*Сигнал=1 *)
   repeat until port(Address) and Bit_weight=0,                   (*Сигнал=0 *)
   time1 =read_8253,                              (*Считывание количества тактов 8253 первый раз *)
   repeat until port(Address) and Bit_weight=Bit_weight,          (*Сигнал снова равен 1 *)
   time2 =read_8253,                              (*Считывание количества тактов 8253 второй раз *)
   Average_number =round(100/(Time1-Time2)), (*Нахождение переменной Average_number *)
   if Average_number=0 then Average_number =1,
   repeat until port(Address) and Bit_weight=Bit_weight,          (*Сигнал=1 *)
   repeat until port(Address) and Bit_weight=0,                   (*Сигнал=0 *)
   time1 =read_8253,                              (*Считывание количества тактов 8253 первый раз *)
   for count =1 to Average_number do              (*Нахождение заднего фронта цифрового сигнала *)
      begin
         repeat until port(Address) and Bit_weight=Bit_weight,    (*Сигнал=1 *)
         repeat until port(Address) and Bit_weight=0,             (*Сигнал=0 *)
      end,
   Time2 =read_8253,                              (*Считывание количества тактов 8253 второй раз *)
   Find_period =((Time1-time2)*1/(2*1193180)*1e6/Average_number),
end,

(*Главная программа *)
begin
   Centronic_address,
   init_8253(255,255)                             (*Инициализация третьего таймера 8253 *)
   repeat
   time_period =find_period(P_address+1, 8)  (*P_address+1 - адрес порта состояния,
                                              8 - вес бита DB3 *)
      gotoxy(15,10)  write( Time period of output signal [ns]  ,time_period 8 1),
      gotoxy(15,11), write( Frequency of the output signal [Hz]. ,1/time period*1e6 8 1),
      delay(5000),
   until keypressed,
end
```

6.3. Цифровые датчики интенсивности света

Оптоэлектронные датчики превращают световой поток в электрический сигнал определенной частоты. С выхода прибора снимается ток или напряжение, пропорциональное интенсивности светового потока. Подобные микросхемы, как правило, содержат усилительные схемы для повышения чувствительности датчика. Если информацию с такого датчика необходимо загрузить в ПК, то между ним и компьютером требуется включить АЦП.

Микросхема TSL220 (Texas Instruments, RS194-278) – это преобразователь интенсивности света в частоту. Внутри корпуса микросхемы находятся кремниевый

фотодиод 4 мм^2 и преобразователь. На выходе формируется последовательность импульсов с частотой, пропорциональной интенсивности света, который падает на фотодиод (рис. 6.25).

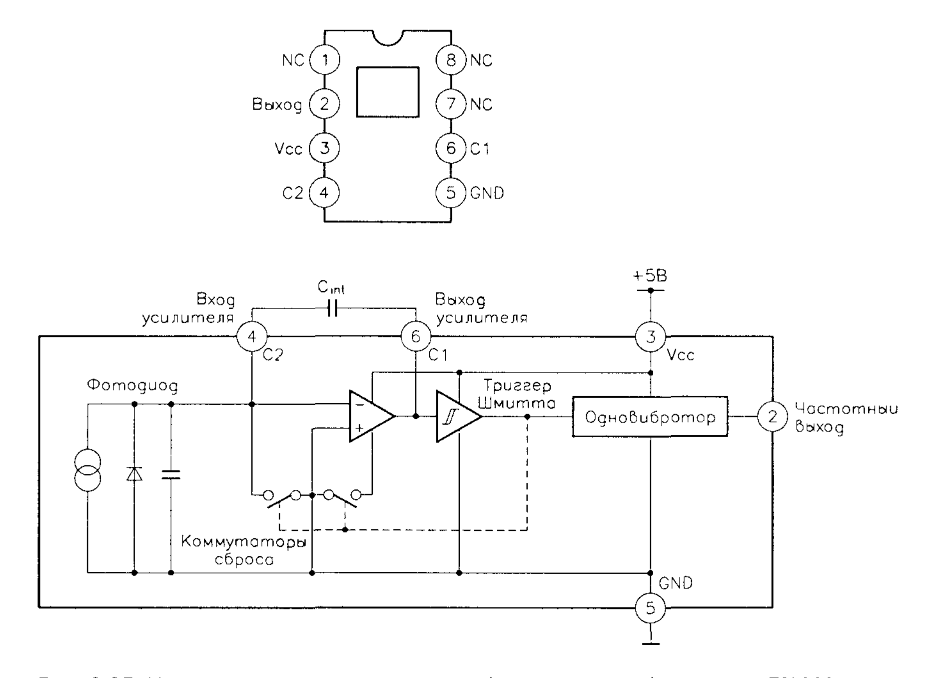


Рис. 6.25. Назначение выводов и внутренняя блок-схема преобразователя TSL220

Преобразователь состоит из интегратора на операционном усилителе (ОУ), транзисторных ключей, триггера Шмитта и одновибратора. Выходная частота микросхемы TSL220 определяется током фотодиода и емкостью интегрирующего конденсатора. Частотный диапазон устройства устанавливается подбором конденсатора. Руководство по выбору номиналов элементов содержится в технической документации изготовителя. Конденсатор соединен с катодом фотодиода (контакт 4) и выходом ОУ (контакт 6). Выходной сигнал совместим по уровню с КМОП схемами. Если устройство соединяется со схемами ТТЛ, то нужно применить нагрузочный резистор 3,3 кОм. Напряжение питания может быть от 4 до 40 В при потребляемом токе 7,5 мА.

Схема на базе этого преобразователя и экспериментальной платы параллельного порта приведена на рис. 6.26. Выход микросхемы TSL220 соединен с контактом S1 на плате. Программа управления измеряет частоту выходного сигнала. Она написана на языке ТР6 и использует встроенный в компьютер таймер/счетчик 8253.

Текст программы TSL220

```
Program TSL220,
(*Программа управления преобразователем свет-частота TSL220 *)
(*Конденсатор 0 01 мкФ, 1% *)
```

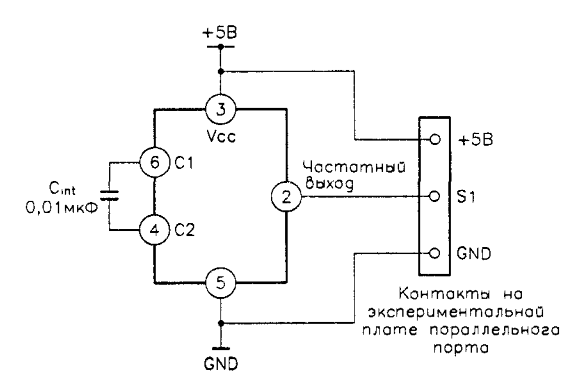


Рис. 6.26. Схема с использованием микросхемы TSL220

```
(*Если интенсивность света слишком мала, эта программа не будет работать.*)
(*Выход TSL220 соединен с контактом S1 экспериментальной платы параллельного порта.*)
uses
  graph,crt,dos;

var
time_period:real;

(*Загрузка файла библиотеки.*)
{$I c:\ioexp\tplib1.pas}

Procedure init_8253(low_count_byte, high_count_byte:byte);
(*Загрузка low_byte и high_byte в третий таймер 8253.*)
(*Тактовая частота 8253: 2×1193180 = 2386360 Гц, период тактовых импульсов 1/f = 0,419 нс.*)
begin
(*Управляющее слово = b6h = 10110110b:
10 = выбор счетчика 2;
11 = чтение/запись сначала младшего байта, а потом старшего;
011 = режим 3;
0 = двоичный счет на 16.*)
  port($43):=$b6;              (*Загрузка управляющего слова в регистр 8253.*)
  port($42):=low_count_byte;   (*Загрузка младшего байта.*)
  port($43):=high_count_byte;  (*Загрузка старшего байта.*)
  port($61):=port($61) or 1;   (*Отключение внутреннего динамика.*)
  port($43):=$80;              (*80H - команда фиксации для третьего счетчика.*)
end;

Function read_8253:integer;
(*Считывание двух восьмибитовых регистров счетчика.*)
var
low_byte,high_byte:byte;
begin
  low_byte:=port($42);
  high_byte:=port($43);
  read_8253:=low_byte+256*high_byte;
end;
```

```
Function find_period(Address:integer;Bit_weight:integer):real;
(*Определение периода входного цифрового сигнала.
Входной сигнал зависит от адреса входного порта (Address) и бита.
Бит 0, Bit_weight=1
Бит 1, Bit_weight=2
...
Бит 7, Bit_weight=128.*)
var
count, average_number,time1,time2:integer;
begin
(*Определение длительности нулевого уровня цифрового сигнала. Она будет использоваться
для установки значения переменной Average_number.*)

  repeat until port(Address) and Bit_weight=Bit_weight;          (*Сигнал=1.*)
  repeat until port(Address) and Bit_weight=0;                   (*Сигнал=0.*)
  time1:=read_8253;                    (*Считывание количества тактов 8253 первый раз.*)
  repeat until port(Address) and Bit_weight=Bit_weight;          (*Сигнал снова равен 1.*)
  time2:=read_8253;                    (*Считывание количества тактов 8253 второй раз.*)
  Average_number:=round(100/(Time1-Time2));         (*Нахождение переменной Average_number.*)
  if Average_number=0 then Average_number:=1;
  repeat until port(Address) and Bit_weight=Bit_weight;          (*Сигнал=1.*)
  repeat until port(Address) and Bit_weight=0;                   (*Сигнал=0.*)
  time1:=read_8253;                    (*Считывание количества тактов 8253 первый раз.*)
  for count:=1 to Average_number do    (*Нахождение отрицательного фронта цифрового сигнала.*)
     begin
        repeat until port(Address) and Bit_weight=Bit_weight;  (*Сигнал=1.*)
        repeat until port(Address) and Bit_weight=0;           (*Сигнал=0.*)
     end;
  Time2:=read_8253;                    (*Считывание количества тактов 8253 второй раз.*)
  Find_period:=((Time1-time2)*1/(2*1193180)*1e6/Average_number);
end;

(*Главная программа.*)
begin
  Centronic_address;
  init_8253(255,255);                             (*Инициализация третьего таймера 8253.*)
  repeat
     time_period:=find_period(P_address+1, 8); (*P_address+1 - адрес порта состояния,
                                                8 - вес бита D83.*)
     gotoxy(10,10); write('Time period of output signal from TSL220 [ns]:',time_period:8:1);
     gotoxy(10,11); write('Frequency of the output signal [Hz]:',1/time period*1e6:8:1);
     delay(2000);
  until keypressed;
end.
```

6.3.1. Линейная матрица световых детекторов TSL215

Оптоэлектронный датчик TSL215 (Texas Instruments) состоит из двух секций по 64 пиксела, организованных в линейную матрицу из 128 пикселов. Пикселы имеют размеры 120×70 мкм и расположены на расстоянии 125 мкм между центрами. Работа датчика подразделяется на два этапа: интегрирование и вывод данных.

На первом этапе каждый пиксел накапливает заряд, пропорциональный интенсивности падающего света; лучше освещаемые участки матрицы получают больший заряд. На втором этапе напряжение заряда каждой точки выводится через аналоговый выход. Это напряжение можно оцифровать с помощью АЦП.

Назначение выводов и внутренняя блок-схема светового детектора TSL215 приведены на рис. 6.27. На контакты 1 и 7, обозначенные Vdd, подается напряжение питания +5 В. Контакты 5 и 12 соединяются с общим проводом. Контакты 4 и 8 – это аналоговые выходы первой и второй секции AO1 и AO2. На вход CLK (контакт 3) подаются тактовые импульсы. Входы SI1 и SI2 (контакты 2 и 10)

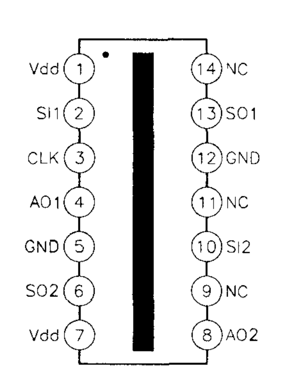


Рис. 6.27. Назначение выводов и внутренняя блок-схема TSL215

используются для управления временем интегрирования и последовательностью вывода точек для двух секций микросхемы. Выходы SO1 и SO2 (контакты 13 и 6) – это последовательные выходы двух секций.

Данное устройство имеет два режима вывода: параллельный и последовательный. В параллельном режиме данные выводятся через выходы AO1 и AO2 под управлением входов CLK, SI1 и SI2. Входы SI1 и SI2 соединены между собой. Для оцифровки двух аналоговых сигналов необходимы два канала АЦП. В последовательном режиме (рис. 6.28) AO1 и AO2 соединены между собой. Сначала выводятся значения напряжений точек первой секции, а затем – второй, при этом необходим всего один АЦП. SO1 соединен с SI2. Внешний сигнал управления поступает на вход SI1. Сначала на вход SI1 подается положительный фронт, после чего этап интегрирования завершается, и начинается этап вывода данных.

Одновременно начинается новый этап интегрирования. Сразу после первого тактового импульса на выходе AO1 появляется напряжение первой точки первой секции. На вход SI1 необходимо подать ноль до прохождения отрицательного фронта тактового импульса. Все остальные значения напряжений точек первой секции выводятся по прохождении еще 63 фронтов тактовых импульсов. По

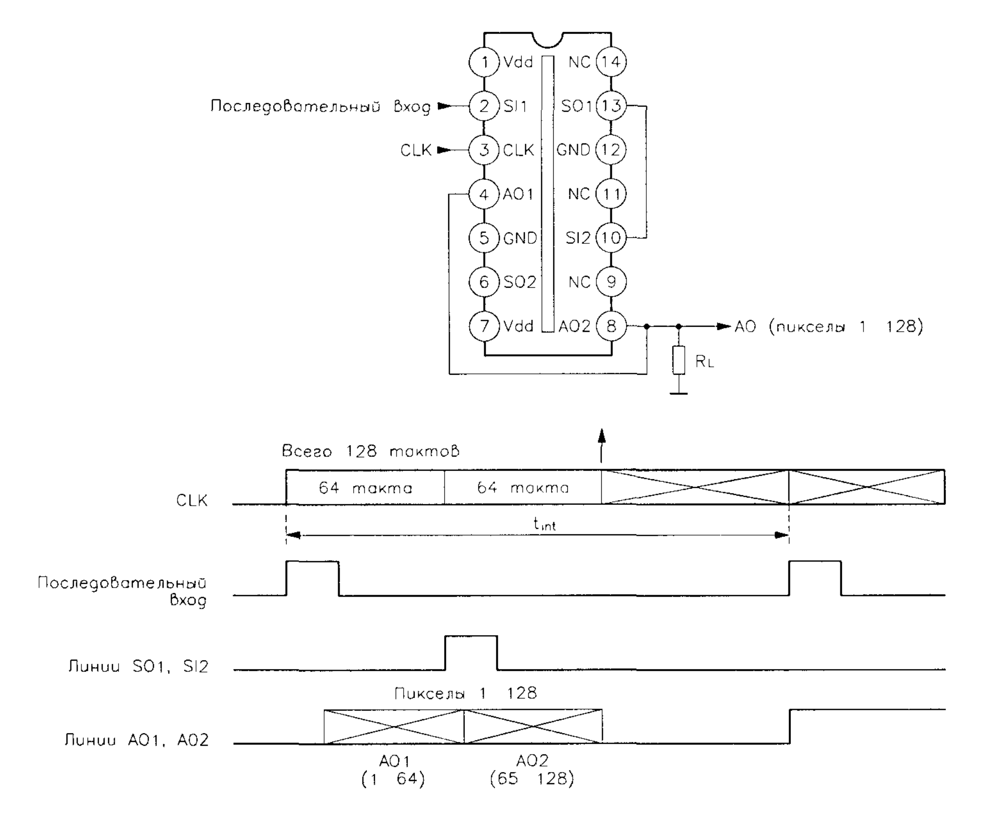


Рис. 6.28. Временные диаграммы последовательного режима вывода данных

фронту 64-го тактового импульса SO1 переходит из нуля в единицу, что переводит SI2 также из нуля в единицу. По фронту 65-го импульса выход AO1 переходит в третье состояние с высоким сопротивлением. В этот момент заканчивается этап интегрирования и начинается этап вывода данных из второй секции. Следующие 64 тактовых импульса выводят значения напряжений точек второй секции. По фронту 129-го импульса выход SO2 переходит в единичное состояние, и AO2 переходит в третье состояние с высоким сопротивлением.

На рис. 6.29 изображена экспериментальная схема, в которой матрица TSL215 настроена на режим последовательного вывода.

Микросхема TSL215 соединена с экспериментальной платой параллельного порта. Вход CLK подключен к контакту D1, вход SI1 – к контакту D2 на плате, выход микросхемы – к осциллографу, отображающему электрический сигнал, который соответствует интенсивности света, воспринимаемого матрицей датчиков. Программа на языке TP6 генерирует сигналы CLK и SI1. Некоторые варианты приложений с использованием этой микросхемы приведены на рис. 6.30.

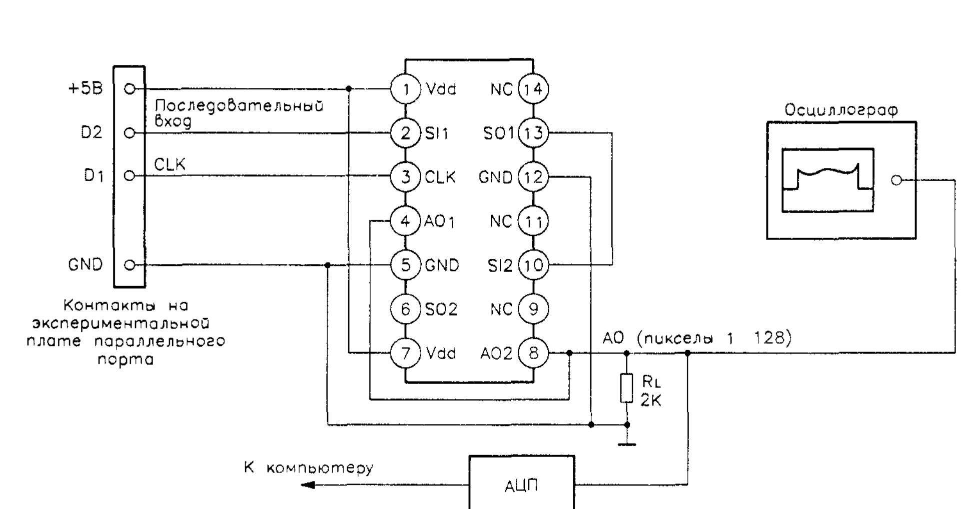


Рис. 6.29. Схема с использованием матрицы TSL215

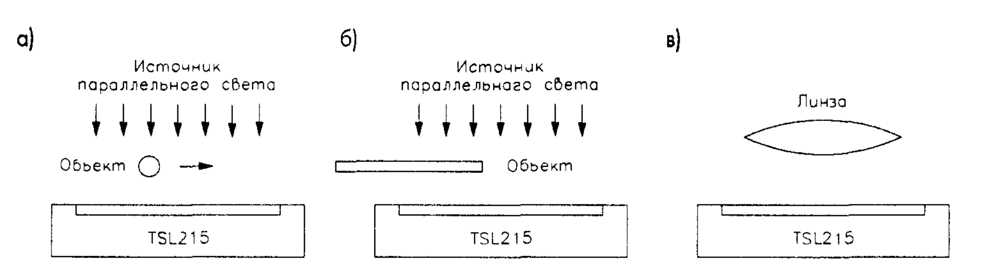


Рис. 6.30. Варианты применения матрицы TSL215 а – обнаружение перемещения, б – обнаружение края предмета, в – обнаружение изображений

Текст программы TSL215.PAS

```
Program TSL215;
(*CLK соединен с D1, SI - с D2.*)
uses
dos,Crt;
var
i,polarity:integer;
dummy:real;

{$I c:\ioexp\tplib1.pas}

Procedure time_delay;
var
ij:integer;
begin
  for ij:=1 to 1 do ij:=ij;
end;

Procedure Read_pixel;
begin
  write_data_port(P_address,0+2);
  write_data_port(P_address,1+2);
for I:=1 to 128 do
  begin
      write_data_port(P_address,0+0);
      time_delay;
      write_data_port(P_address,1+0);
      time_delay;
  end;
end;

(*Главная программа.*)
begin
  Centronic_address;
  Repeat
      read_pixel;
      delay(3);
  until keypressed;
end.
```

6.3.2. Другие цифровые оптоэлектронные датчики

Микросхема IS1U60 (Sharp, RS577-897) – это *инфракрасный приемник* для пультов дистанционного управления (рис. 6.31). Он помещен в пластиковый корпус и содержит схему, способную принимать модулированный инфракрасный сигнал частотой 38 кГц и преобразовывать его в последовательность импульсов. Устройство питается от напряжения +5 В и потребляет ток 3 мА. Цифровые сигналы для надежного приема информации должны иметь длительность высокого и низкого уровней не менее 400–800 мкс. Среднее расстояние срабатывания равно 5 м при угле приема до 30°.

Микросхемы IS485 и IS486 (Sharp, RS197-031) – это *оптические детекторы на триггере Шмитта* (рис. 6.32). Они помещены в пластиковый корпус со встроенным ограничивающим фильтром дневного света. Устройство состоит из фотодиода,

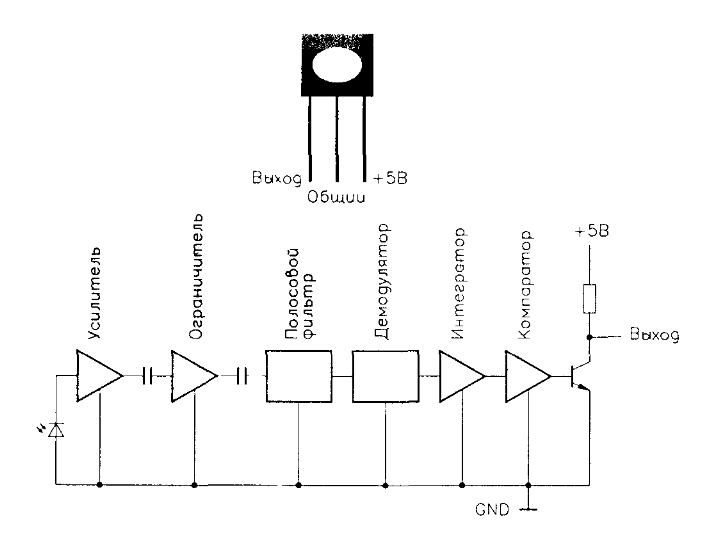


Рис. 6.31. Назначение выводов и внутренняя блок-схема IS1U60

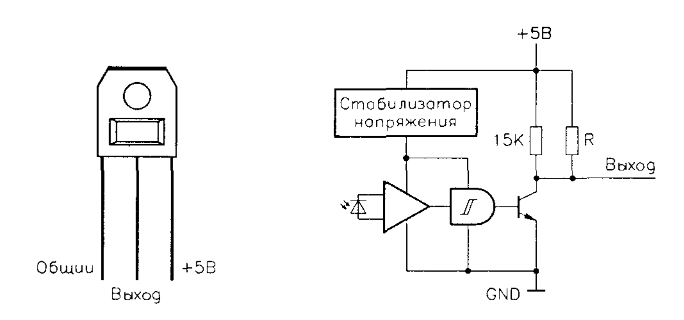


Рис. 6.32. Назначение выводов и внутренняя блок-схема детекторов IS485 и IS486

усилителя, стабилизатора напряжения, триггера Шмитта и усилителя, совместимого с ТТЛ/ТТЛШ и КМОП логикой. Напряжение питания микросхемы может быть от 4,5 до 17 В На выходе необходимо использовать нагрузочный резистор (500 Ом – 50 кОм). При величине нагрузочного резистора в 1 кОм время нарастания выходного импульса составляет 100 нс, время спада – 50 нс, а угол приема равен ±20°.

6.4. Цифровые датчики температуры

Для измерения температуры в цифровых устройствах необходимы температурный датчик, схема управления и АЦП. Последние модели температурных датчиков совмещают эти компоненты в одном кристалле.

6.4.1. Термометр DS1620

Микросхема DS1620 (Dallas, RS218-3810) – это девятиразрядный термометр и термостат, служащий для измерения и отображения температуры (рис. 6.33). Он имеет три выхода, которые используются при работе микросхемы в режиме термостата. Настройки выходного сигнала можно запрограммировать и сохранить во внутренней энергонезависимой памяти. Устройство измеряет температуру от –55 до +125 °С с шагом 0,5 °С, преобразование занимает 1 с.

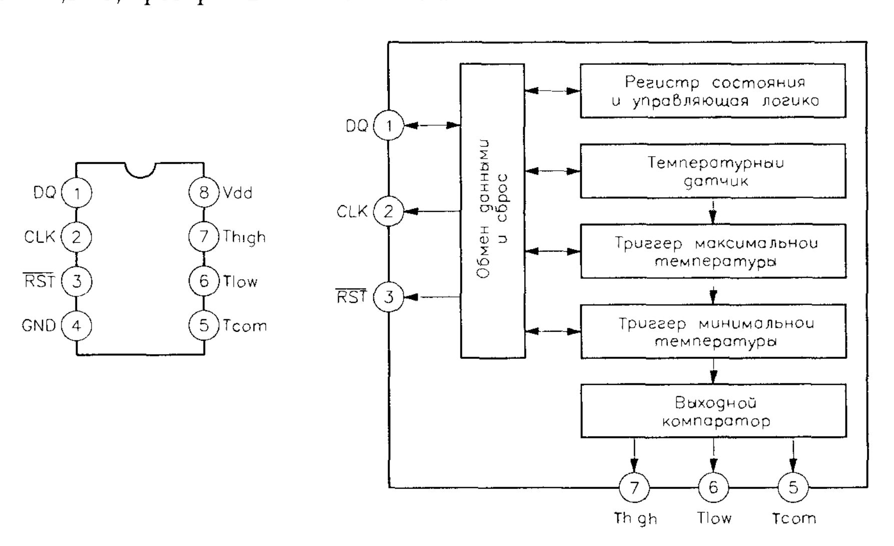


Рис. 6.33. Назначение выводов и внутренняя блок-схема термометра DS1620

Передача данных от микросхемы к внешнему устройству осуществляется по трехпроводной последовательной шине: CLK/$\overline{\text{CONV}}$ (контакт 2), DQ (контакт 1) и RESET (контакт 3). Эти выходы совместимы с уровнями ТТЛ. Thigh (контакт 7) – выход триггера высокой температуры. Если температура превышает установленный верхний порог, то выход Thigh сигнализирует об этом высоким уровнем и остается в таком состоянии до тех пор, пока температура не упадет ниже заданного порога. Tlow (контакт 6) – выход триггера низкой температуры. Если температура опускается ниже определенного нижнего предела, то на нем появляется сигнал высокого уровня, сохраняющийся до тех пор, пока температура не поднимется выше указанного предела. Tcom (контакт 5) – это выход комбинированного триггера высокой и низкой температуры. Tcom = 1, когда температура превышает верхний предел, Tcom = 0, когда она опускается ниже нижнего предела. Контакты 4 и 8 соединены с отрицательным и положительным проводами источника питания. Потребляемый ток в режиме ожидания равен 1 мкА, в рабочем режиме – 1 мА.

Управление устройством осуществляется в два этапа: сначала команды управления последовательно загружаются в микросхему, а затем девятиразрядное число,

соответствующее температуре, либо считывается, либо записывается. Микросхема имеет девять команд:

- Read temp (AAh): чтение значения регистра, содержащего результат последнего измерения, – 9 бит данных;
- Start conversion T (EEh): запуск процесса измерения температуры. Данные не передаются;
- Stop convert T (22h): остановка измерения. Данные не передаются;
- Write TH (01h): запись верхнего предела в триггер высокой температуры – 9 бит данных;
- Write TL (02h): запись нижнего предела в триггер низкой температуры – 9 бит данных;
- Read TH (A1h): чтение содержимого триггера высокой температуры – 9 бит данных;
- Read TL (A2h): чтение содержимого триггера низкой температуры – 9 бит данных;
- Write configuration (0Ch): запись настроечных данных в регистр настройки – 8 бит данных;
- Read configuration (ACh): чтение настроечных данных из регистра настройки – 8 бит данных.

Настроечное слово управляет режимами работы микросхемы DS1620. Оно сохраняется в регистре настройки. Функции битов регистра приведены ниже:

```
DONE THF TLF XXX CPU 1SHOT
```

X	любое
DONE	0 = идет преобразование 1 = преобразование завершено
THF	флаг высокой температуры. Если температура равна или выше верхнего предела, то бит THF = 1. Он остается в единичном состоянии до тех пор, пока его не сбросят, записав ноль, или не отключат питание устройства
TLF	флаг низкой температуры. Если температура равна или ниже нижнего предела, то бит TLF = 1. Он остается в единичном состоянии до тех пор, пока его не сбросят, записав ноль, или не отключат питание устройства
CPU	если CPU = 0, то вход CLK/$\overline{\text{CONV}}$ управляет началом цикла измерения; в противном случае микросхема работает в режиме обмена информацией с внешним устройством
1SHOT	если 1SHOT = 1, микросхема производит один цикл измерения после поступления команды; в противном случае ИС настроена на непрерывное измерение температуры

Данные о температуре имеют девятибитовый формат. Дискретность представления температуры равна 1/2 °С. Некоторые соотношения между значениями температуры и выходными данными приведены ниже:

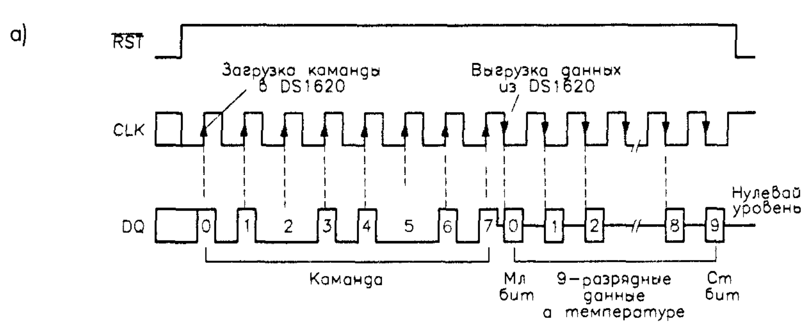
+125 °С 0 11111010 (00FA)
+25 °С 0 00110010 (0032)

+1/2 °C	0 00000001 (0001)
0 °C	0 00000000 (0000)
−1/2 °C	1 11111111 (01FF)
−25 °C	1 11001110 (01CE)
−55 °C	1 10010010 (0192)

Временные диаграммы передачи данных представлены на рис. 6.34. Передача начинается при поступлении положительного фронта на вход $\overline{RST}$ (контакт 3). Если на этот вход подать 0, то передача прекращается. Процессами чтения и записи управляет тактирующий вход микросхемы. Один тактовый цикл состоит из отрицательного фронта и следующего за ним положительного. При записи информации состояние битов данных должно оставаться неизменным во время прохождения положительного фронта. При считывании данные выводятся из устройства по каждому отрицательному фронту тактовых импульсов. Когда на тактовом входе высокий уровень, выход DQ (контакт 1) имеет высокое сопротивление. При чтении данных младший бит передается первым. Через этот контакт можно как принимать, так и передавать данные.

Схема с использованием термометра DS1620, подключенного к экспериментальной плате параллельного порта, приведена на рис. 6.35.

Поскольку вывод DQ может использоваться как для чтения, так и для записи, а на экспериментальной плате параллельного порта таких двунаправленных



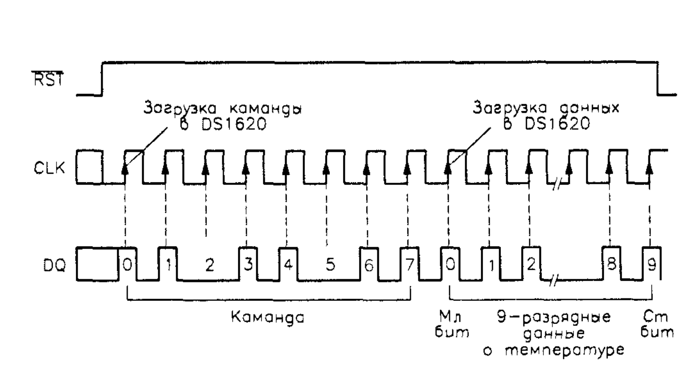


Рис. 6.34. Временные диаграммы работы микросхемы-термометра DS1620 а – последовательность выгрузки данных, б – последовательность загрузки данных

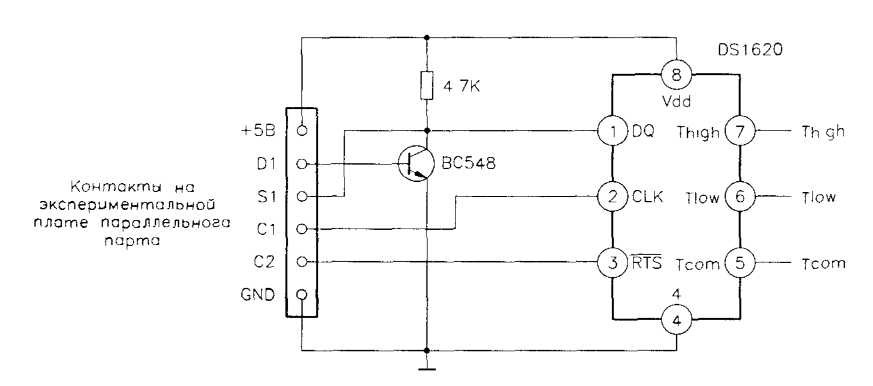


Рис. 6.35. Схема с использованием микросхемы-термометра DS1620

линий нет, необходимо применять транзистор. База транзистора соединена с контактом D1. Когда ИС настроена на прием информации, данные поступают из компьютера на контакт D1, а затем через транзистор в инверсном виде – на микросхему. Когда она передает информацию, транзистор должен быть закрыт (это достигается подачей низкого уровня на его базу через тот же контакт), и данные поступают на контакт S1. Входы CLK/$\overline{\text{CONV}}$ и $\overline{\text{RST}}$ соединены с контактами C1 и C2. После прохождения положительного фронта по входу $\overline{\text{RST}}$ микросхема настраивается на прием управляющей информации. Необходимые данные считываются с линии D1 под управлением тактовых импульсов. Если DS1620 настраивается на вывод данных, то после загрузки в нее управляющей команды D1 переходит в нулевое состояние, а биты данных последовательно выводятся под управлением тактовых импульсов и поступают на контакт S1. Если микросхема должна принимать данные, они также загружаются под управлением тактовых импульсов. Программа на языке TP6 иллюстрирует работу этой ИС.

Текст программы DS1620.PAS

```
Program DS1620_temperature_sensor,
(*Программа управления температурным датчиком DS1620 *)
(*Вход/выход данных DQ соединен с S1 при чтении и с D1 при записи
CLK/CONV соединен с C1  RST - с C2 *)
uses
crt,dos,

{$I c \ioexp\tplib1 pas}

Procedure Write_protocol(datax byte),
(*Запись формата данных в DS1620 *)
var
i byte,
begin
   write_control_port(P_address,1+0),   (*RST=0  CLK=1 *)
   delay(1),
   write_control_port(P_address,1+2),   (*RST становится равным 1 для запуска цикла
                                         ввода/вывода *)
   (*CLK =1 *)
   delay(1)
```

```
  (*Передача битов формата в DS1620.*)
  for I:=1 to 8 do
     begin
        write_control_port(P_address,0+2);
        delay(1);
        write_data_port(P_address,1-round((datax and bit_weight(i))/bit_weight(i)));
        (*Загрузка битов формата в D1 на экспериментальной плате, линия инвертирована.*)
        delay(1);
        write_control_port(P_address,1+2);
        delay(60);
     end;
  delay(50);
end;
Procedure Write_temperature(temp:byte);
(*Запись значения температуры (temp) после записи формата, temp = 0 - 250. Отрицательная
температура не поддерживается.*)
var
i,datax:byte;
begin
  datax:=temp*2;
  (*Передача битов формата в DS1620.*)
  for i:=1 to 9 do
  begin
     write_control_port(P_address,0+2);
     delay(1);
     if i<=8 then write_data_port(P_address,1-round((datax and bit_weight(i))/bit_weight(i)))
     else write_data_port(P_address,0);
     (*Ввод формата данных в D1 на экспериментальной плате, линия инвертирована.*)
     delay(1);
     write_control_port(P_address,1+2);
     delay(60);
  end;
  delay(90);
end;
Function Temperature:real;
var
i,tempx:integer;
bitx:array[1..9] of byte;
begin
  write_data_port(P_address,0);
  for i:=1 to 9 do
     begin
        write_control_port(P_address,0+2);
        delay(1);
        bitx[i]:=read_status_port(P_address) and 1;
        write_control_port(P_address,1+2);
        delay(1);
     end;
  write_control_port(P_address,1+0);
  delay(10);
  tempx:=0;
  for i:=1 to 8 do tempx:=tempx+bit_weight(i)*bitx[i];
  Temperature:=(tempx+0*bitx[9]*256)/2;
end;
Procedure Example1;
```

```
(*Пример показывает, как можно установить температурный предел и считать установленное
значение в компьютер.*)
begin
  writeln('Example 1, TH and TL are loaded with 35 and 25 deg C');
  writeln('Reading TH and TL temperature values from the DS1620');
  write_protocol(12);          (*0Ch = запись команды настройки.*)
  write_protocol(0);           (*00h = режим непрерывного измерения температуры.*)
  write_protocol(1);           (*01h = команда записи верхнего температурного предела.*)
  write_temperature(35);       (*85D = верхний предел в градусах Цельсия.*)
  write_protocol(2);           (*02h = команда записи нижнего температурного предела.*)
  write_temperature(25);       (*20D = нижний предел в градусах Цельсия.*)
  write_protocol(161);         (*A1h = команда считывания содержимого ЦПУ в TH.*)
  writeln('High temperature limit:',temperature:5:1,' °C'); (*Считывание верхнего предела
                                                              температуры.*)
  write_protocol(162);         (*A2h = команда считывания содержимого ЦПУ в TL.*)
  writeln('Low temperature limit.',temperature:5'1,' °C'); (*Считывание нижнего предела
                                                             температуры.*)
end:

Procedure Example2;
(*Проверка других свойств.*)
begin
  write_protocol(12);
  write_protocol(01):      (*Запуск режима однократного измерения.*)
  write_protocol(34);      (*22h = остановка режима однократного измерения.*)
  writeln;
  repeat
      write_protocol(170); (*AAh = команда считывания температуры.*)
      gotoxy(8,10);
      write('DS1620 stopped, Old temperature measured by DS1620 [deg C]: ',temperature:5:1);
      (*После остановки DS1620 считывается старое значение температуры.*)
      delay(5000);
  until keypressed;
  readln;
  write_protocol(238);     (*EEh = начало преобразования температуры.*)
  repeat
      write_protocol(170); (*AAh = команда считывания температуры.*)
      gotoxy(5,12);
      write('DS1620 active, present temperature measured by DS1620 [deg C].
      temperature:5:1);
      (*Многократное измерение нового значения температуры.*)
      delay(5000);
  until keypressed;
end;

(*Главная программа.*)
begin
  centronic_address;
  example1,
  example2;
end.
```

6.4.2. Цифровой температурный датчик

Микросхема SMARTEC (LJK Technology) – это температурный датчик, который позволяет передавать в компьютер информацию о температуре только по одной

линии (рис. 6.36). На выход микросхемы подается ТТЛ/КМОП совместимый прямоугольный сигнал частотой 4 кГц и изменяемой скважностью, зависящей от температуры. По данным измерения скважности такого сигнала можно вычислить значение температуры. Для работы датчика не нужны АЦП, для сопряжения с внешним устройством используется только один провод. Диапазон измеряемой температуры от –45 до +130 °С. Общая погрешность измерения менее 2 °С во всем диапазоне. Напряжение источника питания от 4,75 до 7 В, ток потребления 200 мкА.

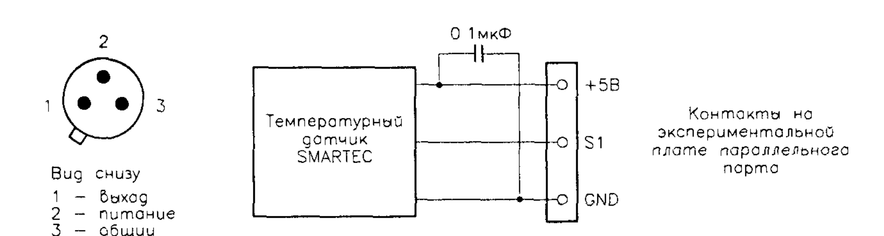


Рис. 6.36. Подключение температурного датчика SMARTEC к экспериментальной плате параллельного порта

Контакт 1 – это выход прямоугольного сигнала, контакты 2 и 3 соединены с положительным и отрицательным проводами источника питания. Скважность линейно зависит от измеряемой температуры и определяется по следующей формуле:

Скважность (%) = 0,320 + 0,00470 × температура (°С).

Выход соединен с одной из линий входного порта ПК. Для вычисления температуры необходимо программно определить длительность временных интервалов высокого и низкого уровней выходного сигнала.

На рис. 6.36 изображено подключение температурного датчика к экспериментальной плате параллельного порта. Программа управления написана на языке TP6.

Текст программы SMT.PAS

```
Program temperature_SMT_sensor
(*Последовательный выход данных (контакт 1) соединен с S1 *)
uses
Crt,dos,

{$I c \ioexp\TPLIB1 PAS}

var
C_or_F,i byte,
datax array[1 12] of byte,
unitx char,

Function temperature real,
(*Вычисление скважности и определение температуры *)
var
lp,hp i total_scan longint,
```

```
begin
  lp =0;                  (*На входе 0, счетчик состояния сбрасывается.*)
  hp:=0;                  (*На входе 1, счетчик состояния сбрасывается.*)
  total_scan.=300000,     (*Общее число опросов.*)
  for i:=1 to Total_scan do
  begin
     if port(P_address+1) and 8=0 then lp:=lp+1,   (*Если вход=0, то low_count=low_count+1 *)
     if port(P_address+1) and 8=8 then hp:=hp+1;   (*Если вход=1,
                                                   то high_count=high_count+1.*)
  end,
  temperature:=(hp/Total_scan-0.32)/0.0047;
end,

(*Главная программа.*)
begin
  Centronic_address;      (*Выбор параллельного порта.*)
  repeat
     gotoxy(10,10);
     write('Temperature from the SMT temperature sensor: ', temperature:5:2,'°C');
  until keypressed;
end
```

6.4.3. Жидкокристаллические температурные модули

Температурный модуль (Maplin FE33L) имеет встроенный температурный датчик и ЖК дисплей, которые установлены на миниатюрной печатной плате с 16-контактным ножевым разъемом. Печатная плата размещена в небольшом пластиковом корпусе вместе с элементом питания напряжением 1,5 В. Модуль не только измеряет и показывает температуру, но и отображает время. Потребление тока – 15 мкА. Некоторые характеристики модуля приведены ниже:

- термометр: 3,5 разряда работают как индикатор градусов Цельсия или Фаренгейта. Диапазон измеряемой температуры от –20 до +70 °С с разрешающей способностью 0,1 °С. Точность ±1 °С в диапазоне 0–40 °С и ±2 °С для других диапазонов. Период дискретизации: выбирается 10 или 1 с. Сигнализирует о самой высокой и самой низкой температурах. При измерении температуры выходные данные выводятся последовательно в двоично-десятичном коде, формат которого представлен на рис. 6.38;
- часы: 3,5 разряда показывают часы и минуты. Точность 0,5 с/сут.;
- источник питания: одна батарея 1,5 В, работающая более года.

Экспериментальная схема изображена на рис. 6.37. Контакты 10 и 9 соединены с контактами S1 и S2 на экспериментальной плате параллельного порта через два транзисторных ключа. Программа на языке TP6, приведенная ниже, считывает последовательные данные и преобразовывает их значение в температуру.

Текст программы TMODULE.PAS

```
Program Temperature_LCD_Module;
(* Выход синхроимпульсов (контакт 10) соединен с S1. Линия инвертирована.
Последовательный выход данных (контакт 9) соединен с S2. Линия инвертирована.*)
uses
crt,dos;
```

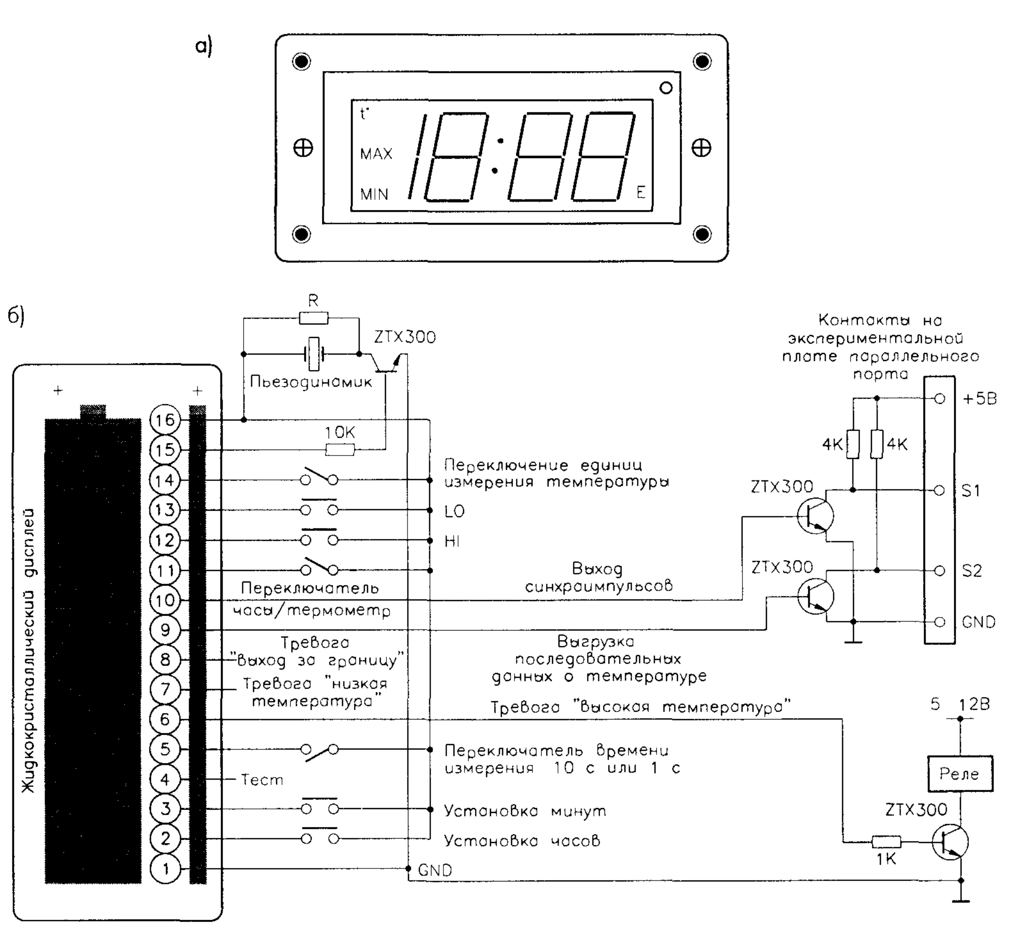
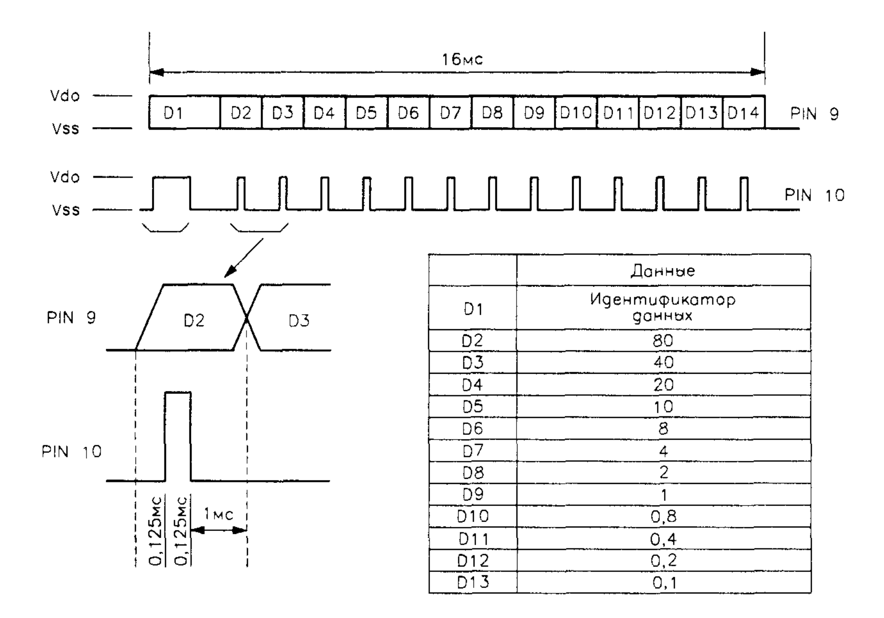


Рис. 6.37. Жидкокристаллический температурный модуль: а – модуль; б – схема включения

```
{$I c \ioexp\tplib1.pas}

var
C_or_F,i:byte,
Datax:array[1..12] of byte,
Unitx char,

Function Temperature: real;
(*Цикл считывания данных с температурного ЖК модуля.*)
begin
  write_data_port(P_address,1);
  (*Нахождение заголовка - кратковременного (1 мс) положительного импульса.*)
  repeat
      repeat until read_status_port(P_address) and 1=1; (*Нахождение логического 0
                                                          Линия инвертирована.*)
      repeat until read_status_port(P_address) and 1=0; (*Нахождение логической 1.
                                                          Линия инвертирована.*)
      delay(3),                                          (*Задержка на 3 мс.*)
```



	Данные
D1	Идентификатор данных
D2	80
D3	40
D4	20
D5	10
D6	8
D7	4
D8	2
D9	1
D10	0,8
D11	0,4
D12	0,2
D13	0,1

Рис. 6.38. Формат последовательных выходных данных

```
    C_or_F:=round(read_status_port(P_address) and 2/2);   (*Получение результата
                                                           в градусах Цельсия (=1)
                                                           или градусах Фаренгейта (=0).*)
    if C_or_F=1 then unitx.='C' else unitx:='F';
  until read_status_port(P_address) and 1=0;

  (*Считывание двоично-десятичных данных.
  Последовательность считывания: 80,40,20,10,8,4,2,1,08,04,02,01.*)
  for i:=1 to 12 do
  begin
    repeat until read_status_port(P_address) and 1=1; (*Нахождение логического 0.*)
    repeat until read_status_port(P_address) and 1=0; (*Нахождение логической 1.*)
    repeat until read_status_port(P_address) and 1=1; (*Нахождение логического 0.*)
    datax(i]:=1-round(read_status_port(P_address) and 2/2);   (*Считывание бита данных.
                                                               Бит инвертирован *)
  end;
  (*Генерация значения температуры.*)
  Temperature:=10*(8*datax[1]+4*datax[2]+2*datax[3]+1*datax[4])+
  1*(8*datax[5]+4*datax[6]+2*datax[7]+1*datax[8])+
  0.1*(8*datax[9]+4*datax[10]+2*datax[11]+1*datax[12]);
end;

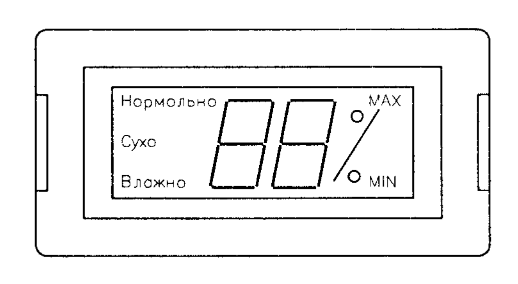
(*Главная программа.*)
begin
  Centronic_address;                              (*Выбор параллельного порта.*)
  repeat
    gotoxy(20,10);
    write('Temperature from the module: ',temperature:5:1, '°',unitx);
    delay(5000);
  until keypressed;
end.
```

6.5. Цифровые датчики влажности

Жидкокристаллический *модуль влажности* (Maplin, ZA38R) имеет такое же устройство, как и модуль температуры/времени, описанный выше. Он измеряет относительную влажность в диапазоне 25–96% и сохраняет минимальный и максимальный результат, полученный после последнего сброса (рис. 6.39). Датчик влажности размещен на плате. Обычно ЖК дисплей показывает текущее значение влажности, но при нажатии кнопок **MIN** и **MAX** на дисплее соответственно отображаются минимальное и максимальное значения, которые хранятся во внутренней памяти датчика. Эту память можно очистить, нажав обе кнопки одновременно.

Для вывода значений относительной влажности в четырехразрядном двоично-десятичном формате используются четыре выхода M1 – M4 (контакты 4–7) – см. рис. 6.40. Сначала по каждой выходной линии передается по два синхроимпульса, затем – четыре бита данных, отображающих состояния кнопок. И только после этого выводятся два разряда значения влажности (старший разряд идет первым).

Схема модуля влажности показана на рис. 6.39. Модуль соединен с экспериментальной платой параллельного порта, четыре выхода – с контактами S1 – S4 на



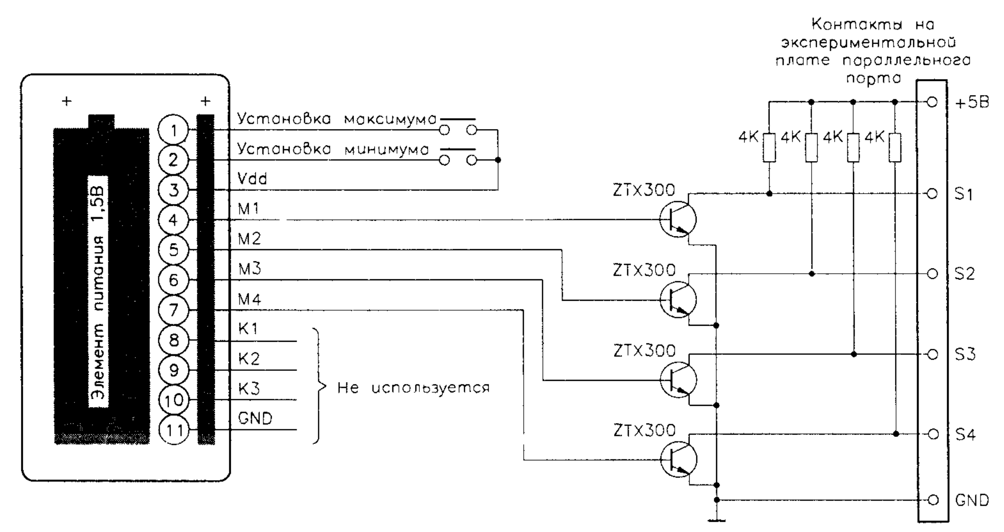


Рис. 6.39. Жидкокристаллический модуль влажности и его включение

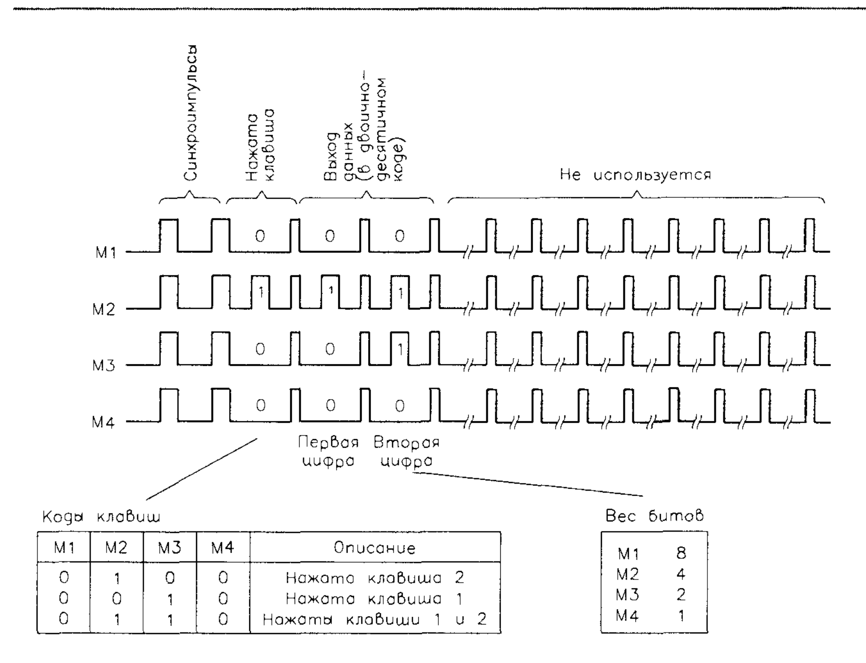


Рис. 6.40. Временные диаграммы выхода последовательных данных

плате через четыре транзисторных ключа. Программа на языке TP6, приведенная ниже, считывает данные из модуля и на их основе вычисляет влажность.

Текст программы HMODULE.PAS

```
Program Humidity_LCD_module,
(* M1 (контакт 4) соединен с S1  Линия инвертирована
   M2 (контакт 5) соединен с S2  Линия инвертирована
   M3 (контакт 6) соединен с S3  Линия инвертирована
   M4 (контакт 7) соединен с S4  Линия инвертирована *)
uses
Crt,dos

{$I c \ioexp\TPLIB1 PAS}

var
C_or_F,i byte,
datax array[1  12] of byte,
unitx char

Function Input_data byte,
(*Все входы порта состояния инвертированы *)
begin
   input_data =5-read_status_port(P_address),
end,

Function Humidity real,
(*Считывание значения влажности из модуля влажности *)
var
keypressed_data digit_1st digit_2nd byte,
```

```
begin
  (*Нахождение логического 0 перед заголовком.*)
  repeat until input_data=15;           (*Нахождение логической 1 на всех входах.*)
  delay(500);                           (*Задержка 500 мс.*)
  (*После задержки на входе 0.*)
  (*Заголовок (два положительных импульса) пропускается.*)
  repeat until input_data=15;           (*Нахождение логической 1.*)
  repeat until input_data=0;            (*Нахождение логического 0.*)
  repeat until input_data=15;           (*Нахождение логической 1.*)
  repeat until input_data=0;            (*Нахождение логического 0.*)

  (*Считывание данных о нажатии клавиши.*)
  repeat until input_data<>0;           (*Ввод данных о нажатии клавиши.*)
  Keypressed_data:=input_data;
  if keypressed_data<15 then
     begin
     (*Пропуск одного синхроимпульса.*)
        repeat until input_data=15;     (*Нахождение логической 1.*)
        repeat until input_data=0;      (*Нахождение логического 0.*)
     end
  else repeat until input_data=0;

  (*Считывание первой цифры.*)
  repeat until input_data<>0;           (*Ввод данных о нажатии клавиши.*)
  digit_1st:=input_data;

  (*Пропуск одного синхроимпульса.*)
  repeat until input_data=15;           (*Нахождение логической 1.*)
  repeat until input_data=0             (*Нахождение логического 0.*)

  (*Считывание второй цифры.*)
  repeat until input_data<>0;           (*Ввод данных о нажатии клавиши.*)
  digit_2nd:=input_data;
  if digit_2nd=15 then digit_2nd:=0;

  (*Генерация значения влажности.*)
  Humidity:=(digit_1st)*10+(digit_2nd);
end;

(*Главная программа.*)
begin
  Centronic_address;   (*Выбор параллельного порта.*)
  Repeat
     gotoxy(20,10);
     write('Humidity from the module:',humidity:5:1,'%');
     delay(5000);
  until keypressed;
end.
```

6.6. Цифровые датчики расхода жидкости

Цифровой датчик расхода жидкости (UCC International, RS185-9982) – это трехконтактный датчик жидкостного потока (рис. 6.41). В его канале расположена нейлоновая крыльчатка, насаженная на вал из нержавеющей стали; когда в канале

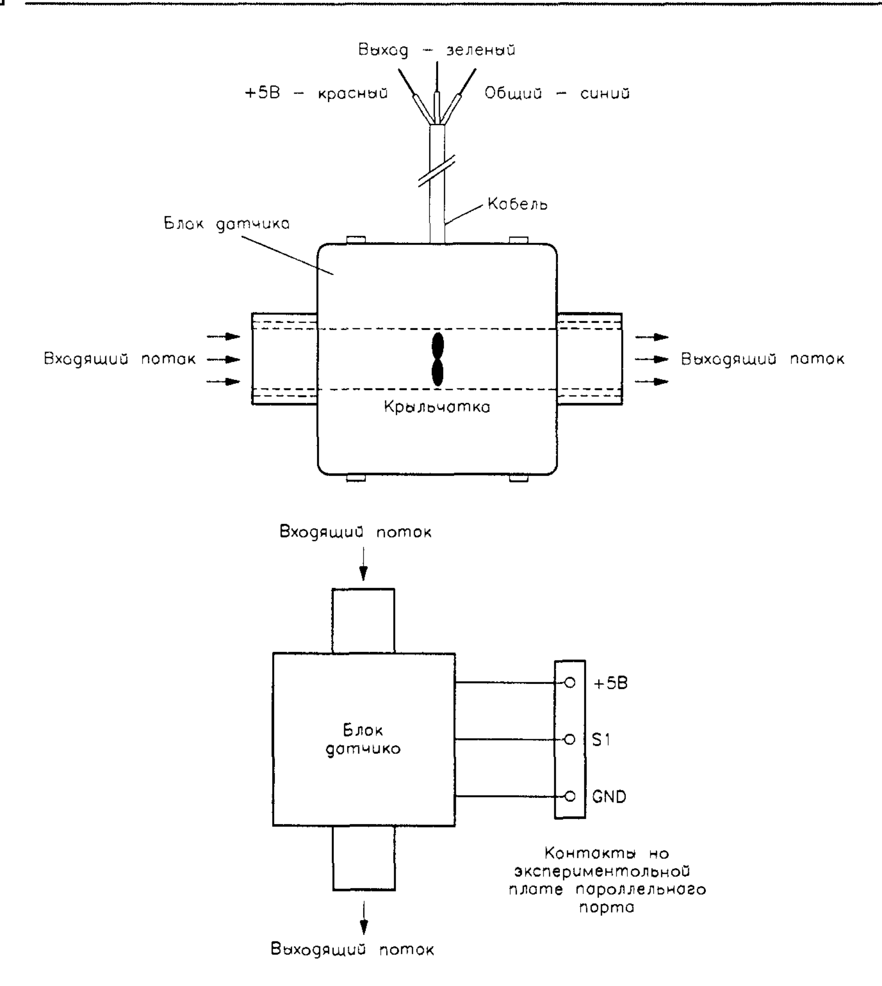


Рис. 6.41. Схемы датчика расхода жидкости

протекает жидкость, крыльчатка вращается. В датчик также встроены инфракрасный излучатель и детектор. Все устройства размещены внутри датчика, внешних элементов не требуется.

Крыльчатка, вращаясь под воздействием потока, периодически закрывает своими лопастями инфракрасный детектор, что приводит к появлению на выходе детектора прямоугольных импульсов, которые после усиления подаются на выходные контакты устройства. Частота следования импульсов пропорциональна скорости потока жидкости. Каждый литр воды, протекающей через датчик, приблизительно соответствует 752 импульсам. Датчик измеряет скорость потока от 1 до 20 л/мин.

На рис. 6.41 изображена схема устройства, в котором датчик соединен с экспериментальной платой параллельного порта. Выходная последовательность импульсов подается на контакт S1 платы. Скорость потока вычисляется по количеству импульсов за установленный промежуток времени Как правило, это делается программно: сначала фиксируется начальное время, а затем подсчитываются импульсы датчика. Когда их число достигает определенного значения, программа снова фиксирует время, после чего вычисляет количество импульсов в секунду и, на основе полученного результата, скорость потока воды.

6.7. Цифровые датчики магнитного поля

В разделе описаны электронные приборы, предназначенные для измерения характеристик магнитного поля.

6.7.1. Цифровой датчик FGM-3 индукции магнитного поля

Чтобы измерить магнитную индукцию посредством цифровых систем, необходимы датчик, схема преобразования и АЦП. Последние разработки в этой области позволили объединить все три устройства в одном корпусе, что значительно упростило процесс измерения и улучшило его качество.

FGM-3 (Speake & Co.) – это трехконтактный датчик, два контакта которого соединены с источником питания, а третий – выход (рис. 6.42). Устройство измеряет индукцию магнитного поля до 10 нТл. На выход подается цифровой сигнал частотой от 50 до 120 кГц. Период сигнала пропорционален величине индукции магнитного поля, воспринимаемого датчиком. FGM-3 имеет низкий температурный коэффициент, равный 0,003% на 1 °С. Высокая чувствительность устройства позволяет измерять индукцию магнитного поля Земли. Датчик можно использовать для ориентирования на местности.

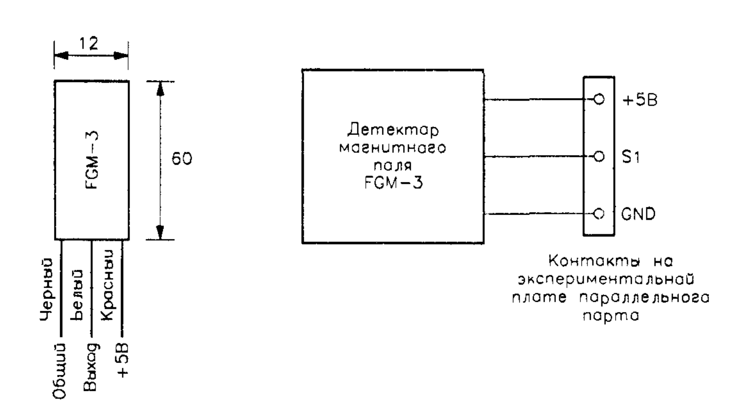


Рис. 6.42. Датчик магнитной индукции FGM-3 и схема его подключения

Напряжение источника питания +5 В. Методика измерения заключается в подсчете количества импульсов за фиксированный промежуток времени. Затем определяется частота сигнала и вычисляется величина индукции магнитного поля.

6.7.2. Цифровой датчик магнитного поля

Микросхема UCN3121 (Allegro, RS307-446) – это интегральный переключатель с открытым коллектором, основанный на эффекте Холла. Он состоит из датчика Холла, усилителя, триггера Шмитта, стабилизатора напряжения и ключа с открытым коллектором (рис. 6.43).

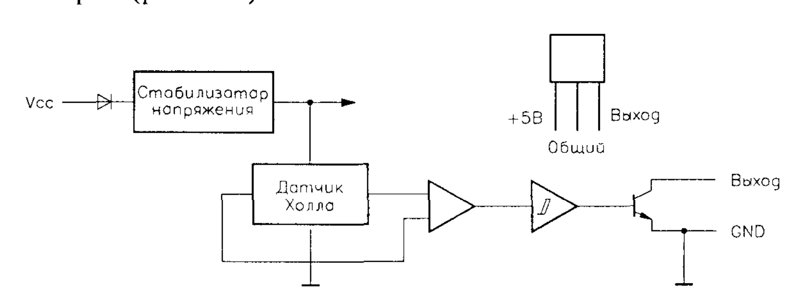


Рис. 6.43. Назначение выводов и внутренняя блок-схема датчика UCN3121

Это пороговый датчик. Когда магнитное поле в датчике Холла превышает порог, на выходе логический ноль. Если магнитное поле ниже порога, на выходе единица. Напряжение источника питания может изменяться от 4,5 до 24 В с номинальным током потребления 5 мА. Выходной каскад – это каскад с открытым коллектором, который выдает ток до 25 мА. Для соединения выхода со схемами ТТЛ/КМОП необходимо между входом питания и выходом подключить нагрузочный резистор номиналом 10 кОм. Более подробные характеристики прибора содержатся в документации изготовителя.

6.8. Радиосистемы точного времени

Приемники радиосигналов точного времени позволяют получить доступ к системе эталонного времени, которая работает с точностью до секунды за миллион лет. Радиосигналы точного времени генерируются Национальной физической лабораторией (NPL) в Англии. Каждую минуту передается двоичный код, содержащий время, дату и день недели. NPL – часть международной сети, которая позволяет сверять часы с точностью до наносекунды в любой точке мира. Позывной передатчика точного времени NPL, расположенного в Регби, – MSF[1].

Передатчик MSF ежеминутно посылает в эфир поток данных, содержащий информацию о времени. В первые семнадцать и последние восемь секунд передается служебная информация, необходимая для синхронизации приемников, в остальные секунды – сведения о времени в двоично-десятичном коде. Формат кода времени MSF приведен в табл. 6.1, а формат передаваемого радиосигнала

[1] Данные об аналогичных российских системах в открытой печати отсутствуют. – *Прим. науч. ред.*

Таблица 6.1. Двоичный код времени передатчика MSF

Секунда	Описание	Двоично-десятичный формат	Описание
0	–		Служебная информация
1	–		Служебная информация
2–16	–		Служебная информация
17	Год (десятки)	80	Год (00–99)
18	Год (десятки)	40	
19	Год (десятки)	20	
20	Год (десятки)	10	
21	Год (единицы)	8	
22	Год (единицы)	4	
23	Год (единицы)	2	
24	Год (единицы)	1	
25	Месяц (десятки)	10	Месяц (01–12)
26	Месяц	8	
27	Месяц	4	
28	Месяц	2	
29	Месяц	1	
30	Число (десятки)	20	Число (01–30)
31	Число (десятки)	10	
32	Число	8	
33	Число	4	
34	Число	2	
35	Число	1	
36	День недели	4	День недели (1–7)
37	День недели	2	
38	День недели	1	
39	Час (десятки)	20	Час (01–24)
40	Час (десятки)	10	
41	Час	8	
42	Час	4	
43	Час	2	
44	Час	1	
45	Минута (десятки)	40	Минута (00–59)
46	Минута (десятки)	20	
47	Минута (десятки)	10	
48	Минута	8	
49	Минута	4	
50	Минута	2	
51	Минута	1	
52	Всегда 0	0	Служебная информация
53–58	Всегда 1	1	Служебная информация
59	Всегда 0	0	Служебная информация

показан на рис. 6.44. Несущая частота равна 60 кГц. Она передается в начале каждой секунды длительностью 100 и 200 мс. Длительность 100 мс соответствует нулю, а 200 мс – единице.

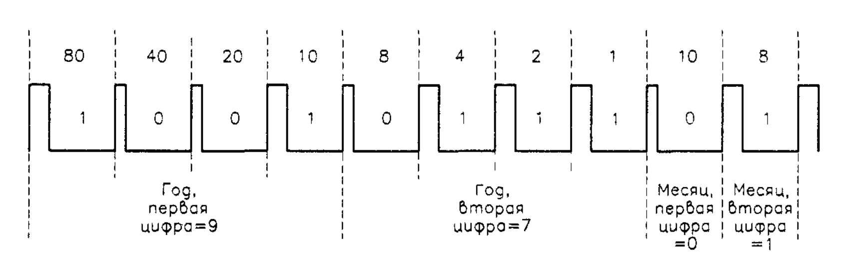


Рис. 6.44. Формат передаваемых данных MSF

Приемный модуль MSF EM2 (Maplin MK68Y) и антенна MSF (Maplin MK72P) составляют приемник кода времени с цифровым последовательным выходом для подключения внешних декодирующих устройств. Приемник обладает высокой чувствительностью при узкой ширине полосы пропускания 10 Гц. Он имеет два режима работы: режим ожидания с малым потреблением мощности и активный режим под управлением входа PON. В первом случае максимальный потребляемый ток равен 1 мкА, во втором – 500 мкА. Напряжение источника питания может быть от 1,5 до 3,5 В. Входы Vcc и GND соединены с положительным и отрицательным проводами источника питания. Для устранения помех в цепи питания используется RC-фильтр первого порядка с сопротивлением резистора 1 кОм и емкостью электролитического конденсатора 10 мкФ. Приемная антенна представляет собой колебательный контур с индуктивностью на ферритовом стержне, она специально разработана для данного приемного модуля.

Схема на базе экспериментальной платы параллельного порта представлена на рис. 6.45а. Цифровой выход MSF-приемника соединен с контактом S1 на экспериментальной плате. Уровень напряжения MSF-приемника приводится к уровням ТТЛ логики с помощью транзисторного ключа. Программа декодера написана на языке TP6.

Текст программы MSF_RES.PAS

```
Program MSF_receiver,
(*Программа управления приемником MSF, выход данных соединен с контактом S1
на экспериментальной плате параллельного порта.*)
uses
crt, dos,

{$I c:\ioexp\tplib1.pas}

Function Period:byte;
(*Определение, что передается каждую секунду, 1 или 0.*)
```

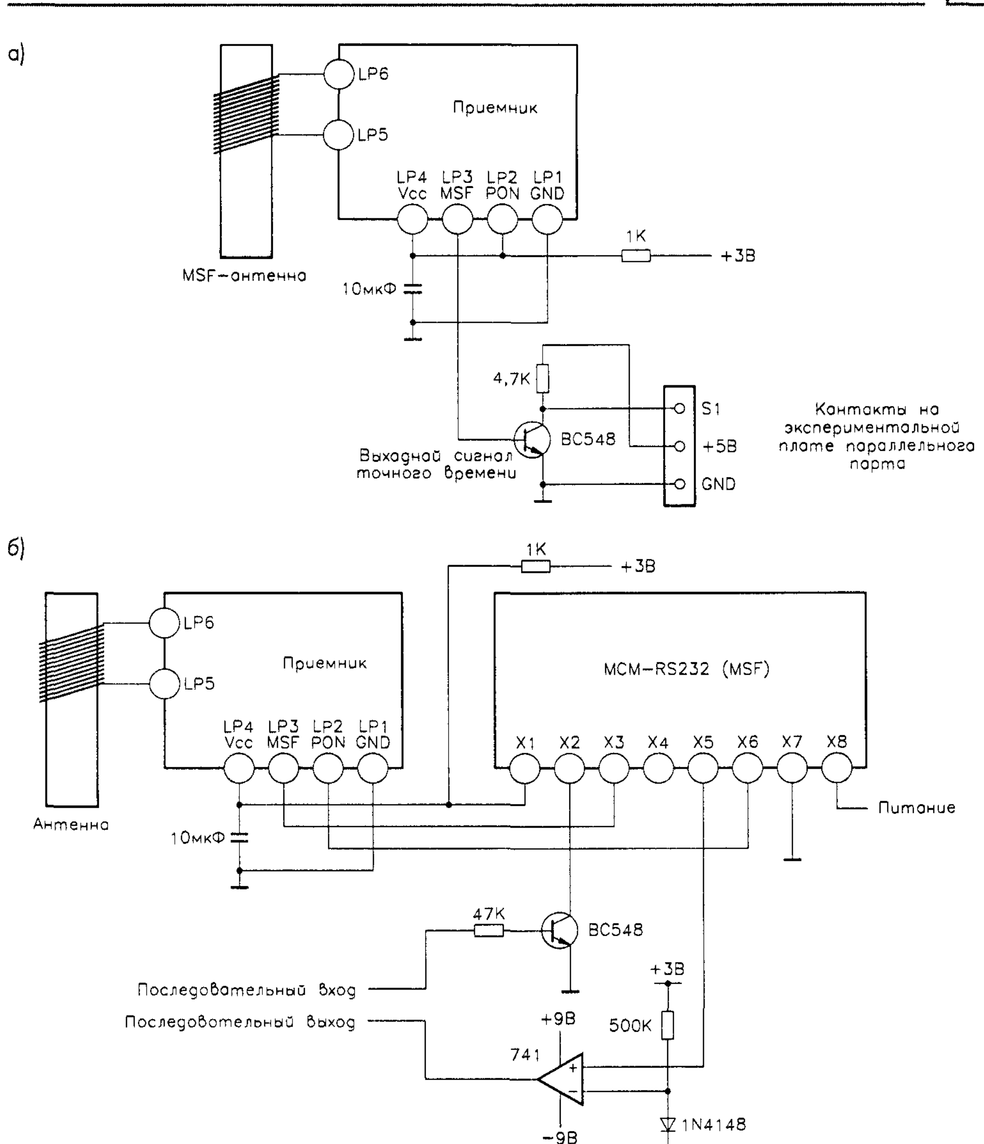


Рис. 6.45. Приложение приемника MSF: а – схема с использованием экспериментальной платы параллельного порта; б – применение модуля MCM-RS232

```
begin
   repeat until read_status_port(P_address) and 1=1,      (*Определение логической 1.*)
   delay(15*11),                                          (*Задержка около 150 мс *)
   if read_status_port(P_address) and 1=1 then period:=1 else period:=0;
   (*Если порт состояния все еще 1, то передавалась 1, если же он равен 0,
   то передаваемые данные тоже должны быть 0.*)
   delay(700*11),                                         (*Задержка около 700 мс.*)
end;
```

```
Procedure get_time_bit;
(*Получение данных о времени.*)
var
count,i:byte;
Year,Month,Day_of_month,Day_of_week,hour,minute:integer;
Tbit:array[1..60] of real;
begin
(*Определение данных, передаваемых между 3 и 16 с (все равны 0).*)
  count:=0;
  repeat
     if period=0 then count:=count+1 else count:=0;
  until Count>=14;

  (*Определение начала передачи данных.*)
  repeat until period=1;
  Tbit[1]:=1;                          (*Данные о годе, бит 3.*)

  (*Получение следующих 35 бит данных.*)
  for i:=2 to 36 do TBit[i]:=period;

  (*Формирование информации о времени.*)
  Year:=round((8*Tbit[1]+4*Tbit[2]+2*Tbit[3]+Tbit[4])*10+(8*Tbit[5]+4*Tbit
  [6]+2*Tbit[7]+Tbit[8]));
  Month:=round(Tbit[9]*10+(8*Tbit[10]+4*Tbit[11]+2*Tbit[12]+Tbit(13]));
  Day_of_Month:=round(10*(2*Tbit[14]+Tbit[15])+(8*Tbit[16]+4*Tbit[17]+2*Tbit[18]+Tbit[19]));
  Day_of_week:=round(4*Tbit[20]+2*Tbit[21]+Tbit[22]);
  hour:=round(10*(2*Tbit[23]+Tbit[24])+(8*Tbit[25]+4*Tbit[26]+2*Tbit[27]+ Tbit[28]));
  Minute:=round(10*(4*Tbit[29]+2*Tbit[30]+Tbit[31])+(8*Tbit[32]+4*Tbit[33]+2*Tbit[34]+Tbit[35]));
  writeln('Year: ',year);
  writeln('Month: ',Month);
  writeln('Day of month: ',Day_of_month);
  writeln('Day_of_week: ',Day_of_week);
  writeln('Hour: ',Hour);
  writeln('Minute: ', minute);
end;

(*Главная программа.*)
begin
  centronic_address;
  repeat
     clrscr;
     get_time_bit;
  until keypressed;
end.
```

Сигнал с выхода MSF-приемника можно подать на микроконтроллерный декодирующий модуль MCM-RS232 (Maplin MK73Q), который выдает информацию о времени в стандартном формате RS232 интерфейса, но с ТТЛ уровнями. Для подключения к ПК требуется внешнее преобразование к двуполярному уровню. Модуль RS232 также непрерывно выдает информацию о точном времени, но компьютеру в этом случае не нужно самостоятельно декодировать данные. Модуль имеет выход управления. Питание на микросхему (вход X8) необязательно подавать постоянно, вход X8 может быть соединен со схемой управления, которая отвечает за

периодическую подстройку времени в соответствующих приложениях. Схема подключения модуля MCM-RS232 приведена на рис. 6.45б. Набор команд для связи с компьютером подробно описан в технической документации изготовителя.

6.9. Клавиатура

Существует два вида *клавиатур*, подключаемых к компьютеру: со сканированием и с кодированием.

Блок-схема 12-клавишной сканирующей клавиатуры показана на рис. 6.46. Клавиши расположены в узлах матрицы, у которой четыре линии строк и три линии столбцов. На линии столбцов по очереди подается отрицательный импульс. В этот момент проверяется состояние четырех линий строк. Если нажатых клавиш нет, все линии строк имеют высокий уровень (они подключены к напряжению +5 В через нагрузочные резисторы). Если же клавиша нажимается, и на линии столбца, соответствующего нажатой клавише, все еще ноль, то адекватная линия строки также становится равной нулю. Зная номера столбца и строки, можно получить позицию нажатой клавиши.

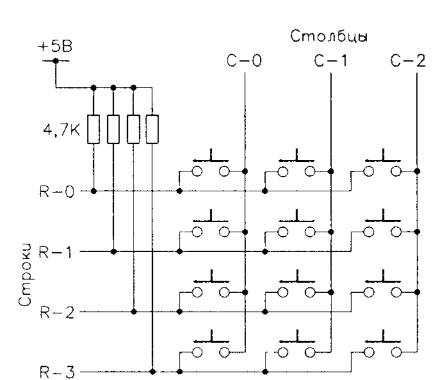


Рис. 6.46. Матричная клавиатура 3×4

В клавиатуре с кодированием использованы специализированные микросхемы, которые обнаруживают нажатие клавиши и показывают ее код в параллельном виде на выходе. Компьютер считывает данные и определяет, какая клавиша нажата. Пример такого устройства – микросхема MM74C922 (National Semiconductors).

7. СОПРЯЖЕНИЕ КОМПЬЮТЕРА С ДРУГИМИ ЦИФРОВЫМИ УСТРОЙСТВАМИ

В данной главе рассказывается, как соединить компьютер с другими устройствами, такими как цифро-аналоговые преобразователи, часы, модули памяти и генераторы сигналов.

7.1. Цифро-аналоговые преобразователи

Цифро-аналоговый преобразователь (ЦАП) – это устройство, которое трансформирует двоичный код в пропорциональное ему аналоговое напряжение или ток. ЦАП широко используется при синтезе речи, музыки и во многих других случаях, когда необходимо обеспечить компьютерное управление внешними аналоговыми устройствами в реальном масштабе времени.

7.1.1. Простой ЦАП R-2R

Простой ЦАП можно построить с помощью резисторной матрицы R-2R, как показано на рис. 7.1. Такое устройство годится для приложений, не требующих большой точности преобразования. Чтобы получить высокое качество преобразования, необходимо использовать специализированные интегральные схемы ЦАП.

7.1.2. ЦАП с параллельным вводом ZN428

Микросхема ZN428 (GEC-Plessey) представляет собой восьмиразрядный цифро-аналоговый преобразователь с буфером-защелкой для входных данных. Он состоит из матрицы R-2R и быстродействующих коммутаторов, что обеспечивает время преобразования, равное 800 нс. Формируемое микросхемой опорное напряжение составляет 2,5 В. Напряжение источника питания +5 В, ток потребления в режиме покоя 20 мА.

На рис. 7.2 приведены расположение выводов и внутренняя блок-схема устройства. Сначала на шину данных подаются 8 бит данных, затем на вход $\overline{\text{ENABLE}}$

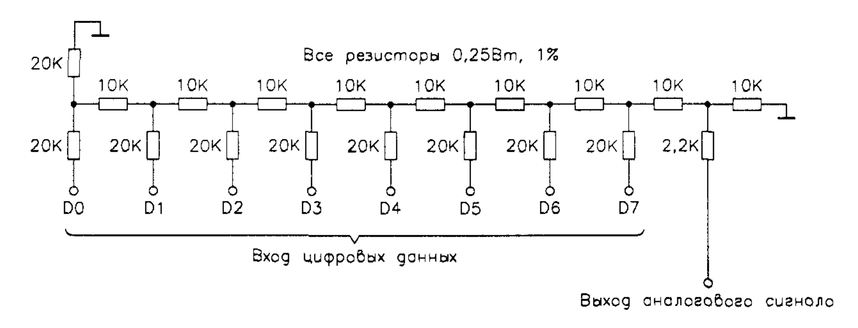
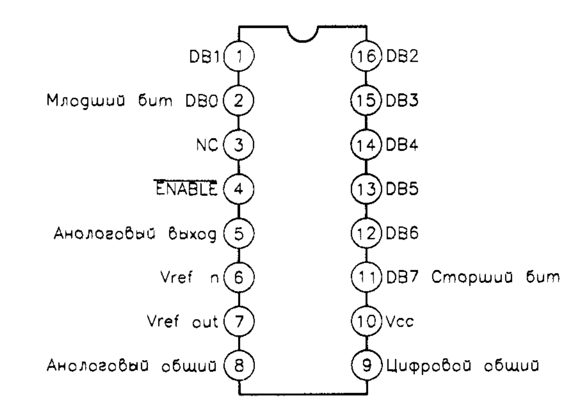


Рис. 7.1. Простая схема ЦАП на основе резисторной матрицы



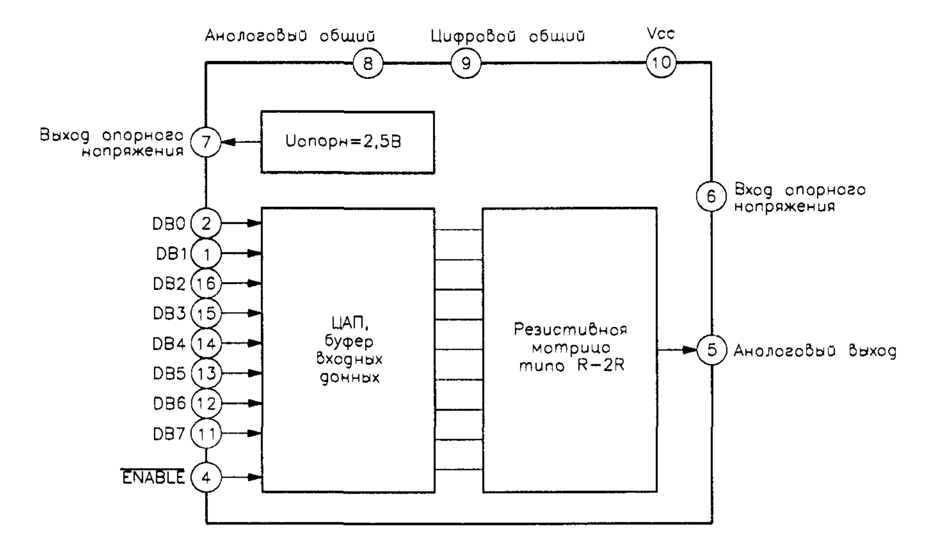


Рис. 7.2. Расположение выводов и внутренняя блок-схема ZN428

(контакт 4) – отрицательный импульс. При этом данные с шины загружаются во внутренний буфер и преобразуются в аналоговое напряжение.

Схема с использованием ЦАП ZN428, подключенная к экспериментальной плате параллельного порта, представлена на рис. 7.3.

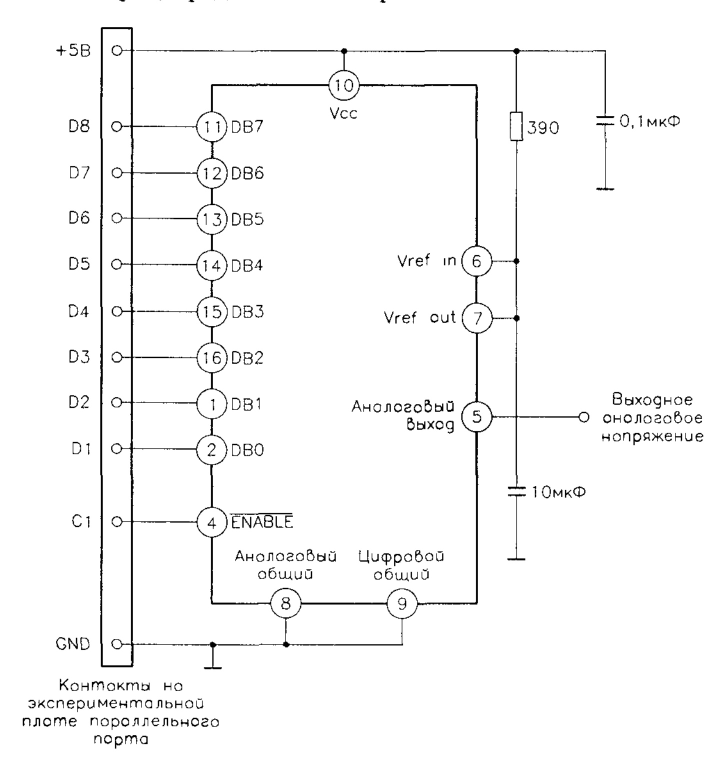


Рис. 7.3. Схема с использованием ЦАП ZN428

Восемь цифровых входов соединены с контактами D1 – D8 платы, вход $\overline{\text{ENABLE}}$ – с контактом C1. Форму аналогового сигнала можно наблюдать при помощи осциллографа. Программа последовательно посылает в ЦАП числа от 0 до 255, вследствие чего на выходе формируется пилообразное напряжение. Демонстрационная программа написана на языке TP6.

Текст программы ZN428.PAS

```
Program ZN428_DAC;
(*Программа управления для ЦАП ZN428.*)
(*Программа демонстрирует, как микросхема генерирует пилообразное напряжение.*)
uses
  crt,dos;
var
  i:integer;
```

```
{$I c:\ioexp\tplib1.pas}

Procedure V_out(data:byte);
(*Вывод двоичного кода для изменения напряжения на выходе.*)
begin
  Write_data_port(P_address,data);     (*Помещение данных (data) в шину данных.*)
  write_control_port(P_address,0);     (*Enable переводится в состояние логического нуля.
                                       Данные записываются в ZN428.*)
  write_control_port(P_address,1);     (*Enable = 1.*)
end;

(*Главная программа.*)
begin
  Centronic_address;
  i:=0;
  repeat
     V_out(i);    (*Входная последовательность 0,1,2,3,...,255 дает на выходе треугольный
                  сигнал.*)
     i:=i+1;      (*i увеличивается на 1.*)
     if i=255 then i:=0;
  until keypressed;
end.
```

7.1.3. ЦАП DAC0854 с последовательным интерфейсом ввода/вывода

Микросхема DAC0854BIN (National Semiconductor, RS853-315) – это четырехканальный восьмиразрядный цифро-аналоговый преобразователь с последовательным интерфейсом ввода/вывода (рис. 7.4).

Напряжение источника питания +5 В, ток потребления 14 мА. Шесть цифровых линий ввода/вывода ($\overline{AU}$, CLK, $\overline{CS}$, $\overline{INT}$, D1 и D0) управляют всеми операциями преобразователя. DAC0854 содержит четыре ЦАП, для каждого из которых имеется вход опорного напряжения (Vref) и выход аналогового напряжения (Vout). В микросхеме есть два входа для подачи напряжения смещения (Vbias 1 и Vbias 2) и вход для подключения источника питания (AVcc). Встроенный источник опорного напряжения формирует напряжение 2,65 В (выход Vref out).

Микросхема DAC0854 способна работать в двух режимах: записи и чтения. В первом случае 8 бит цифровых данных заносятся в ЦАП и преобразовываются в аналоговое напряжение; во втором – данные, записанные в ЦАП, считываются обратно. Запись или чтение могут выполняться одним или всеми ЦАП сразу. Режим устанавливается с помощью управляющего слова, которое помещается в регистр управления. Управляющее слово – это последовательность битов, которая заносится в DAC0854 через вход данных. Функции битов управляющего слова приведены ниже:

бит 1 (стартовый бит)	всегда 1
бит 2 (режим чтения/записи)	0 = RD/$\overline{WR}$ – режим записи
	1 = RD/$\overline{WR}$ – режим чтения

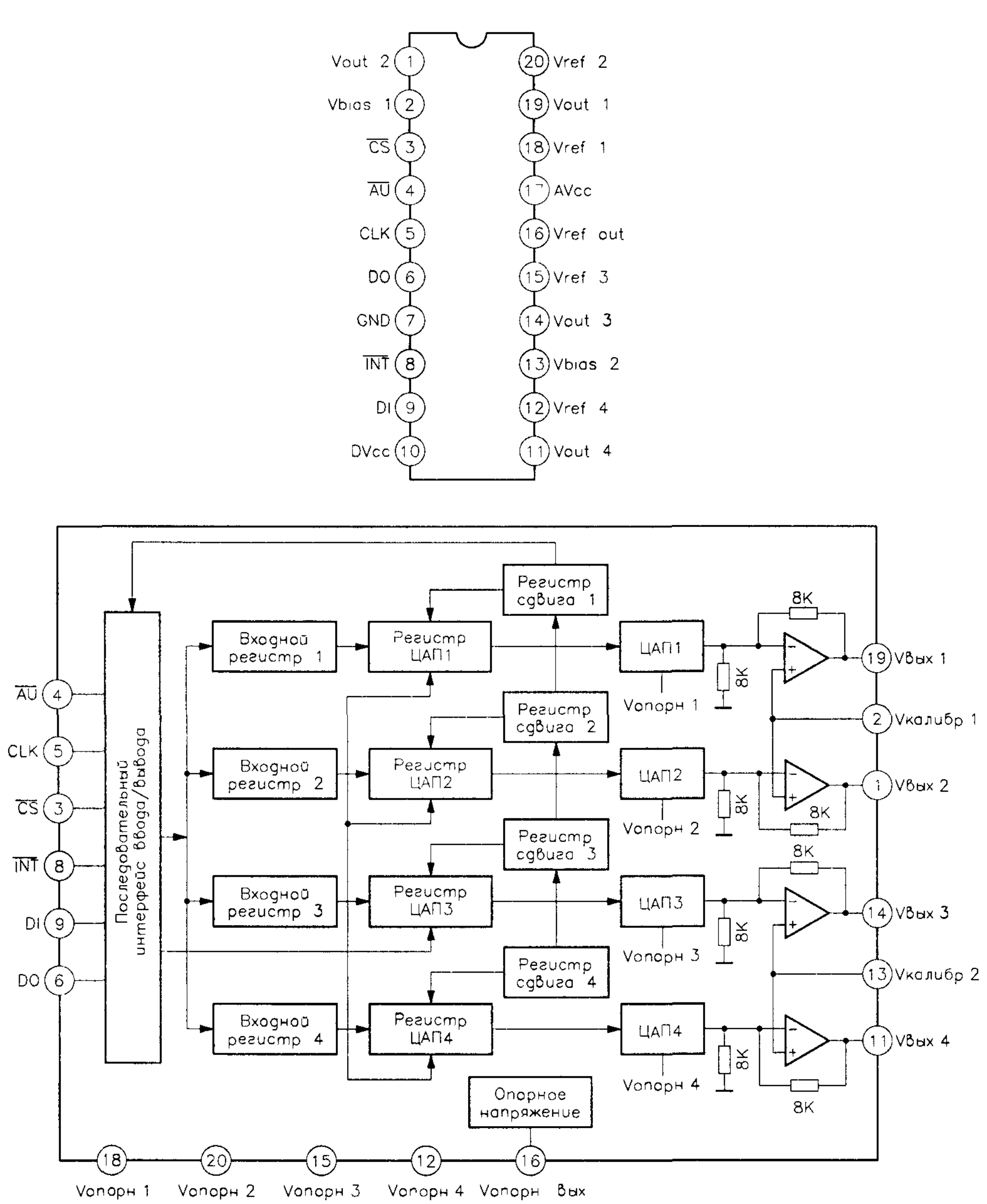


Рис. 7.4. Назначение выводов и внутренняя блок-схема ЦАП DAC0854

бит 3 (глобальная операция)	0 = доступ к одному ЦАП
	1 = доступ ко всем ЦАП
бит 4 (управление обновлением)	0 = нет обновления
	1 = обновление аналогового выхода

бит 5 (адрес) A1 – выбор каналов ЦАП
бит 6 (адрес) A0 – выбор каналов ЦАП

При доступе к одному каналу ЦАП входы A0 и A1 указывают один из четырех каналов. Если выбрана глобальная операция (бит 3 = 1), биты 5 и 6 пропускаются (управляющее слово в таком случае состоит только из четырех битов). Если бит управления обновлением равен 1, то входные цифровые данные преобразовываются в аналоговое напряжение по отрицательному фронту импульса $\overline{CS}$. При этом на входе $\overline{AU}$ (асинхронное обновление) должен быть высокий уровень. Все операции начинаются по отрицательному фронту на входе $\overline{CS}$. Биты управляющего слова поступают на вход D1. Каждый бит передается в ЦАП по положительному фронту тактового импульса. Временные диаграммы режима записи приведены на рис. 7.5.

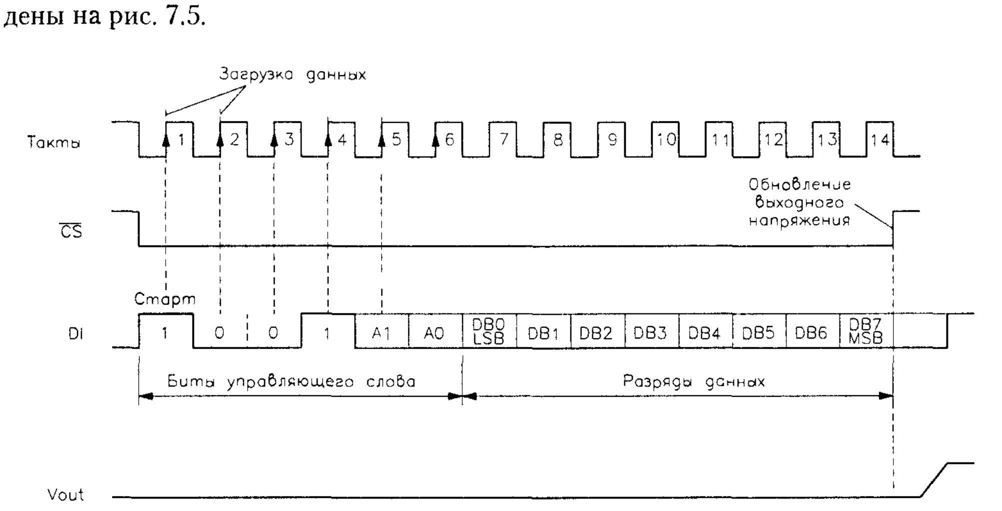


Рис. 7.5. Временные диаграммы режима записи

Диапазон выходного напряжения цифро-аналогового преобразователя может быть различным. Если в качестве опорного напряжения используется внутренний источник на 2,65 В, а для получения напряжения смещения на входе Vbias применяется схема делителя напряжения, то диапазон напряжений составит 0,31–2,81 В. Соотношение между выходным напряжением (Vout) и входными цифровыми данными выражается следующей формулой:

$$Vout = 2{,}500 \frac{DATA}{256} + 0{,}310,$$

где значение DATA представлено в десятичном формате.

Схема подключения ЦАП DAC0854 к экспериментальной плате параллельного порта показана на рис. 7.6. Контакты D1, D2 и D3 платы соединены с входами CLK, $\overline{CS}$ и D1, контакт S1 – с входом D0. Программа управления написана на языке TP6; здесь процедура `ALL_DAC(DATA:byte)` записывает данные (`DATA`) во все ЦАП, а `ONE_DAC(address, DATA:byte)` – в один ЦАП, определенный переменной

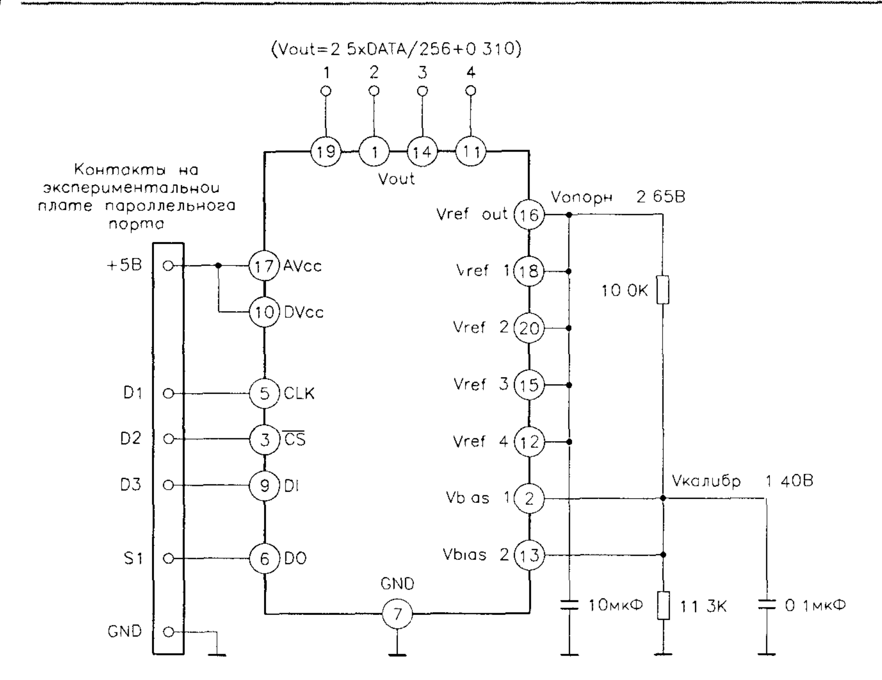


Рис. 7.6. Схема подключения ЦАП DAC0854 к экспериментальной плате параллель ного порта

address Таким образом, микросхема DAC0854 генерирует пилообразный сигнал по всем четырем каналам Форму сигнала можно наблюдать на экране осциллографа

Текст программы DAC0854.PAS

```
Program DAC0854_driver
(*Программа управления четырехканальным ЦАП DAC0854 *)
(*Соединение с экспериментальной платой параллельного порта
  CS - выбор чипа D2
  CLK - такты D1
  DI - вход данных D3
  DO - выход данных S1 *)
uses
  crt dos

{$I c \ioexp\TPLIB1 pas}

var
  command i byte
  D array[1 130] of byte

procedure Init
(*Инициализация линий управления *)
begin
  write_data_port(P_address 1+2+0)     (*Clock=CS=1 *)
end
```

```
Procedure load_data(data·byte);
(*Процедура загрузки данных, CS=0, Data остается постоянным, Clock=переход 0-1 *)
begin
  write_data_port(P_address,0+0+4*data),        (*Вывод данных.*)
  write_data_port(P_address,1+0+4*data);        (*Clock переходит из 0 в 1 для загрузки
                                                данных.*)
  write_data_port(P_address,0+0+0);             (*Clock=0.*)

end;

Procedure All_DAC(data:byte);
var
  i integer;
begin
  load_data(1);   (*Загрузка стартового бита.*)
  load_data(0);   (*Загрузка бита RD/WR=0, операция записи.*)
  load_data(1);   (*Загрузка бита глобальной операции, 1, открытие глобальной операции.*)
  load_data(1);   (*Загрузка бита разрешения обновления, 1, обновление в момент перехода CS
                  из 0 в 1.*)
  (*Загрузка данных.*)
  for i:=1 to 8 do load_data(round(data and bit_weight(i)/bit_weight(i)));
  write_data_port(P_address,0+2+0);     (*Переход CS из 0 в 1 для обновления данных.*)
end,

Procedure One_DAC(address,data.byte);
var
  i·integer;
begin
  load_data(1),   (*Загрузка стартового бита.*)
  load_data(0);   (*Загрузка бита RD/WR=0, операция записи.*)
  load_data(0):   (*Загрузка бита глобальной операции, 0, операция с одним ЦАП *)
  load_data(1);   (*Загрузка бита разрешения обновления, 1, обновление в момент перехода CS
                  из 0 в 1 *)
  load_data(round(address and 2/2)),    (*Загрузка адреса A1.*)
  load_data(address and 1);             (*Загрузка адреса A2 *)
  (*Загрузка данных.*)
  for i:=1 to 8 do load_data(round(data and bit_weight(i)/bit_weight(i))),
  write_data_port(P_address,0+2+0),     (*Переход CS из 0 в 1 для обновления данных.*)
end;

(*Главная программа.*)
begin
  Centronic_address;
  init;
  repeat
      for i:=1 to 255 do one_DAC(3,i),
  until keypressed,
end.
```

7.2. Цифровые потенциометры

Цифровые потенциометры изменяют сопротивление под управлением цифровых схем и используются в качестве цифровых регуляторов громкости в аудиоаппаратуре и усилителях.

Серия X9C103 (Xicor) – это КМОП потенциометры с долговременной памятью (рис. 7.7). К ним относятся несколько устройств: X9C103 (RS299-480), X9C503 (RS299-496) и X9C104 (RS299-503), которые имеют максимальное сопротивление 10, 50 и 100 кОм соответственно. Шаг изменения сопротивления равен максимальному сопротивлению, деленному на 99. Напряжение источника питания +5 В, потребляемый ток 1 мА в режиме регулировки и 0,5 мА в режиме ожидания.

Серия X9C содержит входную секцию управления, счетчик, декодирующую секцию, энергонезависимую память и резисторную матрицу, состоящую из 99 резисторов. Между любыми двумя резисторами на обоих концах матрицы расположены точки отвода, причем каждая точка связана с выводом регулировочного контакта (VW) через транзисторный ключ. Два конца резисторной матрицы VH и VL эквивалентны фиксированным контактам механического потенциометра и могут соединяться с напряжением от –5 до +5 В. VW – это регулировочный контакт, который представляет собой аналог подвижного контакта механического потенциометра.

Положением регулировочного контакта управляют три входа: $\overline{CS}$ (выбор микросхемы), $V/\overline{D}$ (вверх/вниз) и $\overline{INC}$ (увеличение). Вход $V/\overline{D}$ (1 = вверх, 0 = вниз) управляет увеличением или уменьшением значения счетчика. При подаче на вход $\overline{INC}$ отрицательного фронта положение регулировочного контакта изменяется на один шаг вверх или вниз. Подача на вход $\overline{CS}$ сигнала низкого уровня разрешает

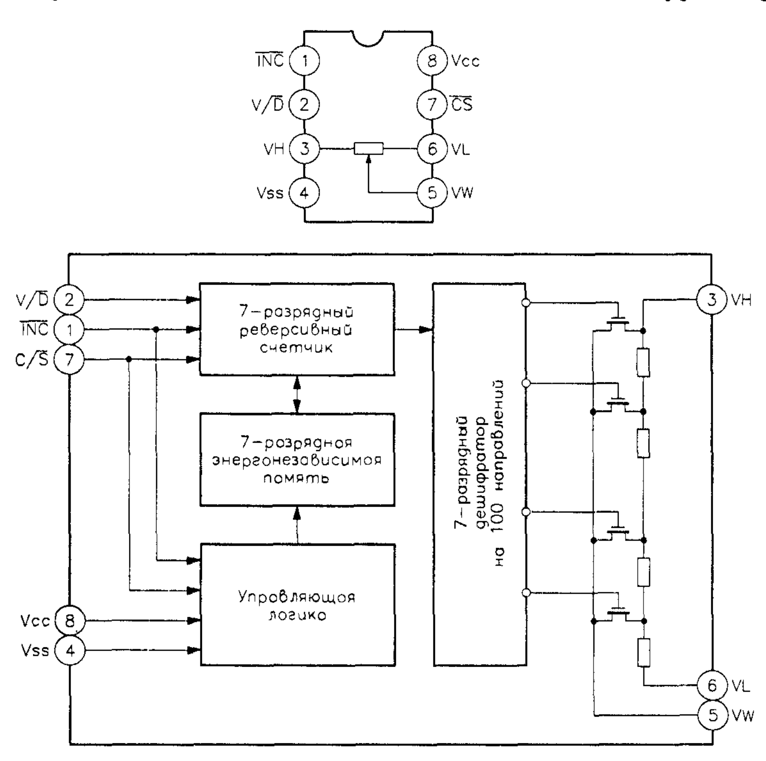


Рис. 7.7. Назначение выводов и внутренняя блок-схема потенциометра серии X9C

доступ к микросхеме. Значение счетчика сохраняется во встроенной энергонезависимой памяти, если сигнал высокого уровня поступит на входы $\overline{CS}$ и $\overline{INC}$. После этого микросхема переходит в режим ожидания с малым потреблением мощности, пока на вход $\overline{CS}$ снова не будет подан сигнал низкого уровня.

Схема подключения потенциометра к экспериментальной плате параллельного порта приведена на рис. 7.8. Входы $\overline{INC}$, $V/\overline{D}$ и $\overline{CS}$ соединены с контактами D1, D2 и D3 на экспериментальной плате. Значение сопротивления между VL и VW измеряется с помощью омметра.

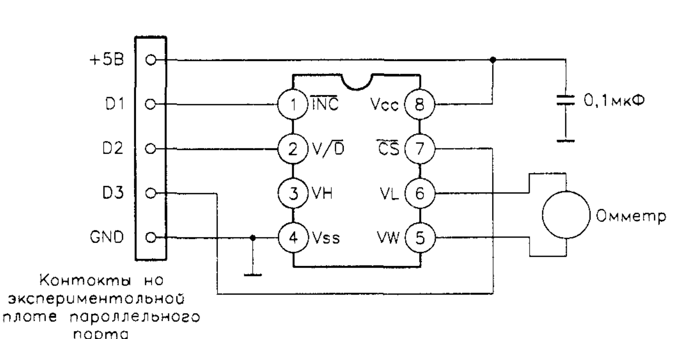


Рис. 7.8. Схема подключения потенциометра X9C103 к экспериментальной плате параллельного порта

Текст программы X9C104.PAS

```
Program X9C104,
(*Программа управления цифровым потенциометром X9C104 *)
(*Соединение с экспериментальной платой параллельного порта:
   INC (увеличение)  D1
   V/D (вверх/вниз): D2
   CS (выбор микросхемы)  D3.*)
uses
   crt,dos;
{$1 c:\ioexp\tplib1.pas}
var
   up_down,step:byte;

Procedure INC_R(step,up_down:byte);
(*Увеличение или уменьшение сопротивления между VL и VW на величину step.*)
(*Направление задается переменной up_down.*)
var
   i:byte;
begin
   for i:=1 to step do
      begin
         write_data_port(P_address,1+2*(up_down)+0);   (*Clock=1.*)
         delay(100);
         write_data_port(P_address,0+2*(up_down)+0),   (*Clock=0 *)
         delay(100);
         write_data_port(P_address,1+2*(up_down)+0),   (*Clock=1.*)
         delay(100);
      end;
```

```
    write_data_port(P_address,1+2*(up_down)+4);   (*Запоминание положения.*)
end;

Procedure test;
begin
    write('Increase [1] or decrease [0] resistance between VL and VW:');
    readln(up_down);
    write('Input steps (1 to 100):');
    readln(step);
    INC_R(step,up_down),
end;

(*Главная программа.*)
begin
    centronic_address,
    repeat
        cerscr,
        test;
        readln;
    until keypressed;
end.
```

7.3. Модули памяти

Модули памяти используются для хранения цифровых данных. RAM – это модули памяти со случайным доступом, позволяющие в любой момент считать или записать данные; если источник питания выключается, данные теряются. ROM – модули памяти только для чтения, перед применением в них необходимо записать информацию. В модули PROM записать информацию можно только один раз, в то время как в стираемые модули EPROM это разрешается делать многократно. Память EPROM по способу стирания информации бывает двух типов: стираемая ультрафиолетовым светом (UVEPROM) и электрически (EEPROM). Модули могут иметь параллельный или последовательный интерфейс ввода/вывода. Параллельный интерфейс состоит из восьми двунаправленных линий данных, нескольких линий адреса и линий управления. Для последовательного интерфейса требуются только три линии управления и одна выходная линия. Кристаллы с шиной I²C имеют две линии ввода/вывода. Модули памяти с шиной MicroLAN (Dallas Touch Memories) используют лишь одну линию.

7.3.1. Модуль EEPROM объемом 2 Кб с последовательным вводом/выводом ST93C56C

Микросхема ST93C56C (SGS-Thomson) – это долговременная КМОП EEPROM объемом 2048 бит с последовательным вводом/выводом (рис. 7.9). Ее память может быть организована двумя способами: 128 слов по 16 разрядов или 256 слов по 8 разрядов в каждом. Любая ячейка памяти стирается и записывается до 1 млн раз.

Vcc (контакт 8) и Vss (контакт 5) соединены с положительным и отрицательным проводами источника питания +5 В. Номинальный ток потребления 2 мА в активном режиме и 50 мкА в режиме ожидания. Контакт 1 – это вход выбора микросхемы, контакт 3 – вход данных, а контакт 2 – вход тактового сигнала. Существует семь команд для управления операциями чтения/записи:

- READ (чтение);
- WRITE (запись);
- EWEN (разрешение стирания и записи);
- EWDS (запрещение стирания и записи);
- ERASE (стирание);
- ERAL (стирание всего содержимого);
- WRAL (запись всего объема информации).

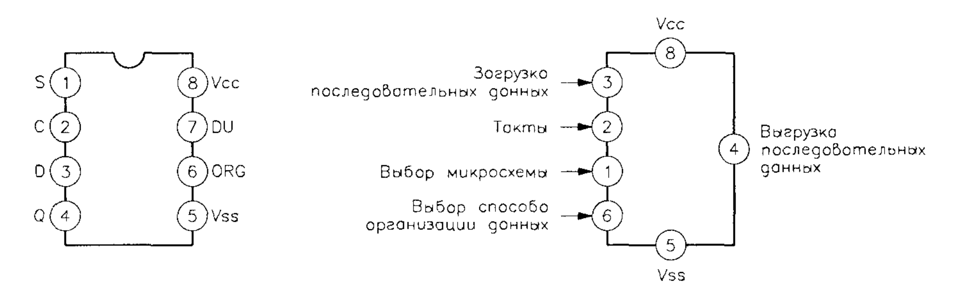


Рис. 7.9. Назначение выводов и логическая схема ST93C56

Рабочий цикл состоит из трех или четырех шагов. Временные диаграммы приведены на рис. 7.10.

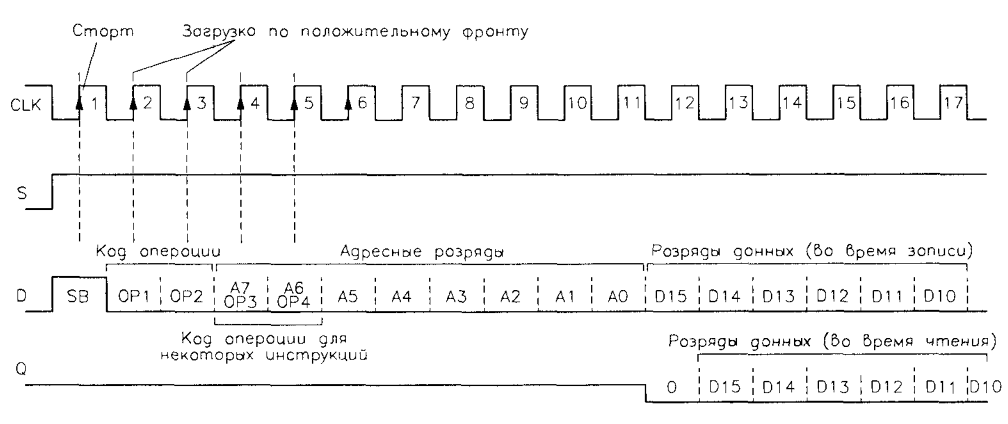


Рис. 7.10. Временные диаграммы работы микросхемы ST93C56C

На первом шаге генерируется начальное условие, которое заставляет ПЗУ реагировать на поступающие команды. Оно предусматривает подачу высокого уровня на вход выбора микросхемы, стартового бита на вход данных и положительного фронта на тактовый вход микросхемы. Затем следует двухбитовый код, который определяет выполняемую операцию. На следующем шаге в микросхему передаются биты адреса и на последнем – данные. Загрузка битов происходит по положительному фронту тактового импульса. Биты адреса заносятся в микросхему, начиная со старшего (шаг 3). На шаге 4 выполняются только операции чтения

и записи данных. Биты данных (сначала старший бит) записываются или считываются по положительному фронту тактового импульса. Набор команд приведен в табл. 7.1.

Таблица 7.1. Команды управления операциями чтения/записи

Команда	Код операции	Адрес	Данные
READ	10	A7 – A0	D15 – D0
WRITE	01	A7 – A0	D15 – D0
EWEN	00	11XXXXXX	нет
EWDS	00	00XXXXXX	нет
ERASE	11	A7 – A0	нет
ERAL	00	10XXXXXX	нет
WRAL	00	01XXXXXX	D15 – D0

Микросхема ST93C56C соединена с экспериментальной платой параллельного порта, как показано на рис. 7.11. Контакты D1, D2 и D3 подключены к S, C и D, контакт S1 – к Q. Организация памяти – 128 слов по 16 разрядов (вывод ORG не подключен). Текст программы управления на языке TP6 приведен ниже.

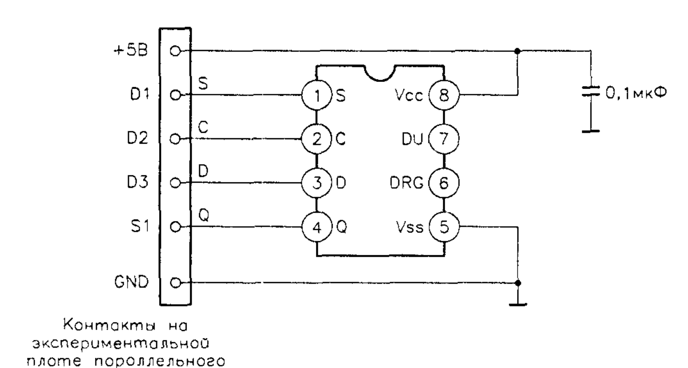


Рис. 7.11. Схема подключения микросхемы ST93C56C к экспериментальной плате параллельного порта

Текст программы 9356.PAS

```
Program EEPROM_93C56_driver;
(*Программа управления EEPROM, организация памяти 128x16.*)
(*Соединение с экспериментальной платой параллельного порта:
  S - выбор микросхемы D2,
  C - такты D1,
  D - вход данных  D3,
  Q - выход данных  S1.*)
uses
  crt,dos,

{$I c \ioexp\TPLIB1 pas}
```

```
var
  command.byte,
  D:array[1..130] of byte;

procedure Init,
(*На все входы управления подан низкий уровень *)
begin
  write_data_port(P_address,0+0+0),
end;

Procedure load_data(data:byte);
(*Процедура загрузки данных, Select=1, Data=постоянные, Clock=переход 0-1 *)
begin
  write_data_port(P_address,0+2+4*data);        (*Select и Data=1.*)
  delay(1);
  write_data_port(P_address,1+2+4*data);        (*Clock переходит из 0 в 1
                                                 для загрузки данных.*)
  delay(1),
  write_data_port(P_address,0+2);               (*Select=1, Clock=Data=0.*)
  delay(1);
end;

Procedure Start;
(*Генерация условия начала. Select=1, Data=1, Clock=переход 0-1.*)
begin
  write_data_port(P_address,0+0+0); delay(1),   (*Select, Clock и Data=0.*)
  write_data_port(P_address,0+2+4), delay(1);   (*Select, Data=1; Clock=0 *)
  write_data_port(P_address,1+2+4), delay(1),   (*Все входы равны 1.*)
  write_data_port(P_address,0+2+0); delay(1),   (*Select=1; Data и Clock=0.*)
end,

Procedure Erase(enable_flag:boolean);
(*Команды разрешения и запрета стирания.*)
var
  i:byte,
begin
  Start;              (*Генерация условия начала *)
  load_data(0);       (*Загрузка первого бита кода операции 0.*)
  load_data(0);       (*Загрузка второго бита кода операции 0 *)
  if enable_flag then
  (*Загрузка 1 и 1 - команда разрешения стирания *)
      begin
          load_data(1); load_data(1);
      end
      else
      (*Загрузка 0 и 0 - команда запрета стирания.*)

      begin
          load_data(1); load_data(1);
      end;

      for i:=1 to 6 do load_data(0),   (*Загрузка шести холостых бит адреса.*)
      delay(1);
      write_data_port(P_address,0);    (*Все входы=0 для блокирования команды *)
      delay(10);
end,
```

```
Procedure Erase_all;
(*Стирание всех ячеек памяти, запись во все ячейки 1.*)
var
  i:byte;
begin
  Start;             (*Генерация условия начала.*)
  load_data(0);      (*Загрузка первого бита кода операции 0.*)
  load_data(0);      (*Загрузка второго бита кода операции 0.*)
  load_data(1);      (*Загрузка 1 и 0.*)
  load_data(0);
  for i:=1 to 6 do load_data(0);      (*Загрузка шести холостых бит адреса.*)
  write_data_port(P_address,0+0+0);         (*Select=0.*)
  delay(1);
  write_data_port(P_address,0+2+0);    (*Select=1 для проверки состояния.*)
  delay(1);
  repeat until read_status_port(P_address) and 1=1; (*Ожидание состояния готовности.*)
  write_data_port(P_address,0+0+0);    (*Select=0 для прекращения выполнения операции.*)
end;

Procedure write_all(byte_low, byte_high:byte);
(*Запись во все ячейки памяти byte_high и byte_low.*)
var
  i:byte;
begin
  Start;             (*Генерация условия начала.*)
  load_data(0);      (*Загрузка первого бита кода операции 0.*)
  load_data(0);      (*Загрузка второго бита кода операции 0.*)
  load_data(0);      (*Загрузка 0 и 1.*)
  load_data(1);
  for i:=1 to 6 do load_data(0);   (*Загрузка шести холостых бит адреса.*)
  (*Загрузка byte_high, сначала старший бит.*)
  for i:=8 downto 1 do load_data(round((byte_high and bit_weight(i))/bit_weight(i)));
  (*Загрузка byte_low, сначала старший бит.*)
  for i:=8 downto 1 do load_data(round((byte_low and bit_weight(i))/bit_weight(i))) ;
  write_data_port(P_address,0+0+0);
  write_data_port(P_address,0+2+0);
  repeat until read_status_port(P_address) and 1=1;
  write_data_port(P_address,0+0+0);
end;

Procedure WriteROM(address,byte_low,byte_high:byte);
(*Запись byte_high и byte_low в две восьмиразрядные ячейки памяти.*)
var
  i:byte;
begin
  Start;             (*Генерация условия начала.*)
  load_data(0);      (*Загрузка первого бита кода операции 0.*)
  load_data(1);      (*Загрузка второго бита кода операции 1.*)
  (*Загрузка адреса.*)
  for i:=8 downto 1 do
     begin
        load_data(round(address and bit_weight(i)/bit_weight(i)));
     end; (*Загрузка адреса 0.*);
  (*Загрузка byte_high, сначала старший бит.*)
  for i:=8 downto 1 do load_data(round((byte_high and bit_weight(i))/bit_weight(i)));
```

```
    (*Загрузка byte_low, сначала старший бит.*)
    for i:=8 downto 1 do load_data(round((byte_low and bit_weight(i))/bit_weight(i)));
    write_data_port(P_address,0+0+0);
    write_data_port(P_address,0+2+0);
    repeat until read_status_port(P_address) and 1=1;
    write_data_port(P_address,0+0+0);
end;

Function readROM(address,x:byte):byte;
(*Считывание двух байтов из 128 ячеек памяти. x=0 считывает младший байт, x=1 - старший байт.*)
var
    i,high_byte,low_byte,dummy:byte;
begin
    start;              (*Генерация условия начала.*)
    load_data(1);       (*Загрузка кода операции 1.*)
    load_data(0);       (*Загрузка кода операции 0.*)
    (*Загрузка адреса.*)
    for i:=8 downto 1 do
        begin
            load_data(round(address and bit_weight(i)/bit_weight(i)));
        end;
    (*Загрузка адреса 0.*);
    dummy:=0;           (*Начало считывания старшего байта.*)
    for i:=8 downto 1 do
        begin
            write_data_port(P_address,0+2); delay(1); (*Clock=0.Select=1.*)
            write_data_port(P_address,1+2); delay(1); (*Clock=1.Select=1.*)
            dummy:=dummy+read_status_port(P_address) and 1*bit_weight(i);
                        (*Считывание бита данных, сначала старший бит.*)
            write_data_port(P_address,0+2);
        end;
    high_byte:=dummy;
    dummy:=0;           (*Начало считывания младшего байта.*)
    for i:=8 downto 1 do
        begin
            write_data_port(P_address,0+2); delay(1);
            write_data_port(P_address,1+2); delay(1);
            dummy:=dummy+read_status_port(P_address) and 1*bit_weight(i);
            write_data_port(P_address,0+2);
        end;
        low_byte:=dummy;

        (*Установка значения данных.*)
        if x=0 then readROM:=low_byte;
        if x=1 then readROM:=high_byte;
end;

Procedure Program_ROM;
(*Тестовая программа.*)
var
    strx;string[250];
    i:byte;
begin
    writeln('This is the message originally stored in the EPROM');
    writeln;
```

```
        for i:=1 to 125 do
            begin
                write(chr(readROM(i,0)),chr(readROM(i,1)));
            end;
    writeln;
    write('Press RETURN to continue...'); readln;

    for i:=1 to 250 do strx[i]:=' ';
    writeln('Input the message which will be programmed into the eeprom: ');
    readln(strx);
    for i:=1 to 125 do writeROM(i,ord(strx[2*i-1]),ord(strx[2*i]));
    writeln('The EEPROM has been programmed ');
    writeln('This is the message stored in the EPROM now');
    for i:=1 to 125 do
        begin
            write(chr(readROM(i,0)),chr(readROM(i,1)));
        end;
        readln;
end;

(*Главная программа.*)
begin
    Centronic_address;
    init;
    erase(true);
    program_rom;
end.
```

7.3.2. EEPROM с шиной I²C

Микросхема 24LC16B (Microchip, аналог X24C16P, Xicor, RS125-1401) – это EEPROM объемом 16 Кбит с шиной I²C (см. главу 4). Микросхема выполняет функции ведомого устройства. Память организована как 2048 ячеек по 8 разрядов каждая; ячейки можно стирать и снова записывать в них информацию до 1 млн раз. Напряжение источника питания от 2,5 до 5 В, потребляемый ток 1 мА в активном режиме и 10 мкА в режиме ожидания.

Выводы A0, A1 и A2 не используются. Вход WP обеспечивает защиту от записи (высоким уровнем). Обычно для разрешения записи он соединяется с «землей». SCL и SDA – это линии тактового сигнала и данных I²C (рис. 7.12).

Данные записываются и считываются по шине I²C. Операция записи имеет два режима: побайтовой и постраничной записи. В первом случае в ячейку памяти записывается один байт, во втором – в модуль записывается 256 байт за одно обращение. При чтении доступны режим чтения текущего адреса и случайный режим чтения.

После прохождения условия «СТАРТ» в микросхему заносится восьмиразрядный ведомый адрес от ведущего передатчика. Биты адреса (начиная со старшего) равны: 1, 0, 1, 0, B2, B1, B0 и $R/\overline{W}$. Биты с 7 по 4 – это постоянный адрес микросхемы, биты B2, B1 и B0 определяют один из четырех блоков памяти, а бит $R/\overline{W}$ устанавливает, какая операция будет проводиться: для операции чтения $R/\overline{W} = 1$, для операции записи $R/\overline{W} = 0$. После передачи ведомого адреса в микросхему

A0 (не используется) 1
A1 (не используется) 2
A2 (не используется) 3
Vss 4
8 Vcc
7 WP
6 SCL
5 SDA

WP 7
SDA 5
SCL 6
Управления вводом/выводом
Управление хранением информации
X-декодер
Защита данных
EEPROM-матрица
Буфер страницы
Y-декодер
Управление чтением/записью
Vcc 8
Vss 4

Рис. 7.12. Назначение выводов и внутренняя блок-схема модуля памяти 24LC16B

посылается адресный байт, который определяет местоположение ячейки памяти в выбранном блоке. Его значение может быть от 0 до 255. Если проводится операция записи, то следом за ней в микросхему записывается восемь бит данных. В режиме случайного чтения после записи адресного указателя снова генерируется условие «СТАРТ», а потом передаются биты адреса и бит $R/\overline{W}$, установленный в 1 (для режима чтения). Затем данные, хранящиеся в памяти, считываются бит за битом. Временные диаграммы операций чтения и записи представлены на рис. 7.13.

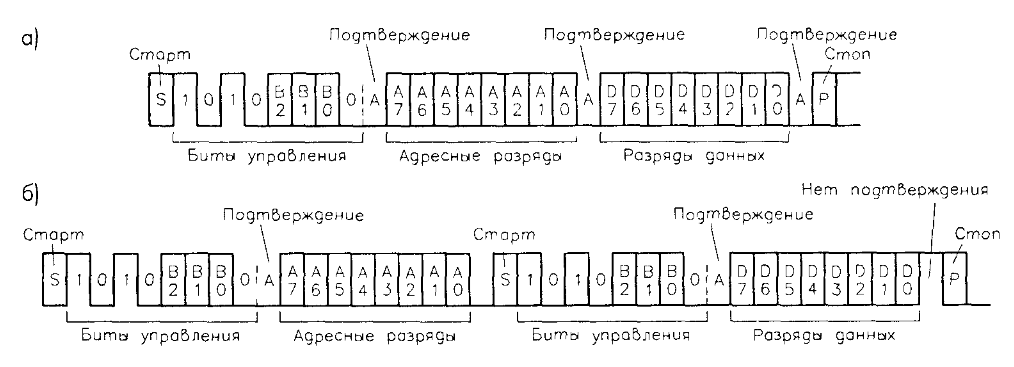


Рис. 7.13. Временные диаграммы операций чтения и записи а – при записи байта, б – при случайном доступе

На рис. 7.14 приведена схема модуля 24LC16B, соединенного с экспериментальной платой параллельного порта. Контакт D1 управляет линией SDA, данные с которой считываются через контакт S1. Контакт C1 управляет линией SCL.

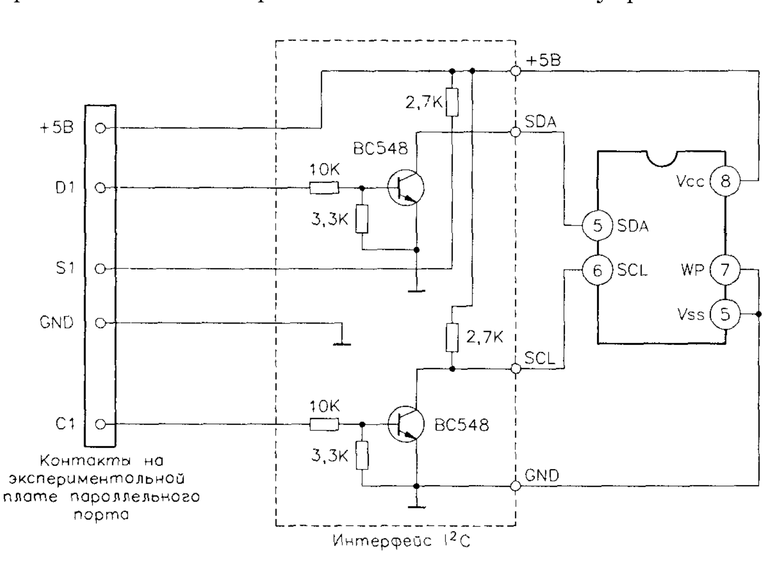


Рис. 7.14. Экспериментальная схема 24LC16B

Текст программы 2416.PAS на TP6

```
Program IIC_memory;
(*Программа управления модулем памяти 24LC16B.*)
(*Компьютер выступает в качестве ведущего устройства, другие микросхемы ведомые *)
(*Соединение с экспериментальной платой параллельного порта.
  D1=SDA, C1=SCK, S1=SDA.*)
(*Условия шины I2C:
  SCL=1, SDA=1:            шина не занята
  SCL=1, SDA=1 - 0:        условие "СТАРТ"
  SCL=1, SDA=0 - 1:        условие "СТОП"
  допустимые данные:       данные постоянны, когда SCL=1
  изменение данных:        когда SCL=0.*)
uses
  crt,dos;
  {$I c:\ioexp\tplib1.pas}
var
  i,ij:integer;
  block,address,data:byte;

(*TP6 программная библиотека I2C № 1, преобразование в двоично-десятичный формат *)
Function BCD(data:byte):byte;
(*Преобразование двоичного кода в двоично-десятичный *)
begin
  BCD:=round((data div 10)*16+10*frac(data/10));
end;
```

```
(*TP6 программная библиотека I²C № 2, запись данных в линию SDA.*)
Procedure SDA(data:byte);
(*Помещение данных в линию SDA.*)
begin
  write_data_port(P_address,1-data)
end;

(*TP6 программная библиотека I²C № 3, управление линией SCL.*)
Procedure SCL(data:byte);
(*Помещение данных в линию SCL.*)

begin
  write_control_port(P_address,1-data);
end;

(*TP6 программная библиотека I²C № 4, инициализация шины I²C.*)
Procedure INIT,
(*Генерация условия инициализации, SDA=SCL=1.*)
begin
  SDA(1);
  SCL(1);
  delay(100),
end;

(*TP6 программная библиотека I²C № 5, генерация условия "СТАРТ".*)
Procedure START;
(*Генерация условия "СТАРТ".*)
begin
  SDA(1); SCL(1); SDA(0); SCL(0);
end;

(*TP6 программная библиотека I²C № 6, генерация тактовых импульсов для подтверждения.*)
Procedure ACK,
(*Генерация подтверждения и соответствующих тактовых импульсов.*)
begin
  SCL(1); SCL(0);
end;

(*TP6 программная библиотека I²C № 7, передача данных в шину I²C.*)
Procedure TRANSMIT(data:byte);
(*Передача данных через шину.*)
var
  i:byte,
begin
  for i:=8 downto 1 do
     begin
        SDA(round(data and bit_weight(i)/bit_weight(i))); (*Помещение данных в линию SDA,
                                                           когда SCL=0.*)
        SCL(1);      (*SCL=0 - 1 *)
        SCL(0);      (*SCL=1 - 0.*)
     end,
  SDA(1),      (*SDA становится равным 1.*)
  ACK;         (*Генерация тактовых импульсов подтверждения.*)
end;
```

```
(*TP6 программная библиотека I²C № 8, прием данных от шины I²C.*)
Function receive(stop_flag:boolean);byte;
(*Прием данных через шину.*)
var
  i,dummy:byte;
begin
  dummy:=0;
  for i:=8 downto 1 do
     begin
        SCL(1);      (*SCL становится равным 1.*)
        delay(1);
        dummy:=dummy+(read_status_port(P_address) and 1)*bit_weight(i);
        (*Считывание данных с линии SDA.*)
        delay(1);
        SCL(0);      (*SCL=1 - 0.*)
     end;
     if stop_flag then
     begin
        SDA(1);      (*Если приняты последние данные, то SDA=1.*)
        ACK;
     end
     else
     begin
        SDA(0);      (*Если принятые данные еще не последние, то SDA=0 для подтверждения.*)
        ACK;         (*Генерация тактовых импульсов подтверждения.*)
        SDA(1);
     end;
  receive:=dummy;
end;

(*TP6 программная библиотека I²C № 9, генерация условия "СТОП".*)
Procedure STOP;
(*Генерация условия "СТОП".*)
begin
  SDA(0);     (*SDA=0.*)
  SCL(1);     (*SCL=1.*)
  SDA(1);     (*SDA=0 - 1.*)
end;

Procedure Write_ROM_byte (block,Address,data:byte);
(*Запись байта только для чтения, block и address определяют ячейку памяти.*)
begin
  start;                     (*Генерация условия "СТАРТ".*)
  transmit(128+32+2*block);  (*Передача битов управления, адреса блока и RD/WR
                             (=0, настройка режима записи).*)
  transmit(address);         (*Передача адреса ячейки памяти внутри блока.*)
  transmit(data);            (*Передача данных.*)
  stop;                      (*Генерация условия окончания.*)
  delay(50);
end;

Function Read_ROM_byte(block,Address:byte):byte;
begin
  start;  (*Генерация условия "СТАРТ".*)
```

```
    transmit(128+32+2*block);        (*Передача битов управления, адреса блока и RD/WR
                                      (=0, настройка режима записи).*)
    transmit(address);               (*Передача адреса ячейки памяти внутри блока.*)
    start;                           (*Генерация условия "СТАРТ".*)
    transmit(128+32+2*block+1);      (*Передача битов управления, адреса блока и RD/WR
                                      (=1, настройка режима чтения).*)
    Read_ROM_byte:=receive(true);    (*Прием данных.*)
    stop;                            (*Генерация условия "СТОП".*)
  end;

  Procedure test_write_read;
  begin
    write('Select a memory block (0,1,2 and 3):'); readln(block);
    write('Select an address in the block (0-255):'); readln(address);
    write('Input the data to be written to ROM:'); readln(data);
    write_ROM_byte(block,address,data);
    writeln;
    writeln('The data written to the ROM:',read_ROM_byte(block,address));
    writeln('Press one RETURN to continue and two RETURN to stop');
    readln;
    delay(4000);
  end;

  (*Главная программа.*)
  begin
    centronic_address;
    repeat
        cerscr;
        Test_write_read;
    until keypressed;
  end.
```

7.4. Системы отсчета реального времени

Микросхемы отсчета времени – обязательный компонент систем управления внешними объектами в реальном масштабе времени. Такие микросхемы, как правило, состоят из автономной схемы отсчета времени/даты и интерфейсных схем сопряжения, которое обычно осуществляется через параллельную шину, имеющую восемь разрядов для данных и несколько линий управления. В качестве примера можно привести микросхемы HD146818 и MSM6242. В последнее время появились новые типы ИС этого класса, например MK41T56 и PCF8573, поддерживающие работу с шиной I^2C. Количество линий ввода/вывода в них существенно меньше.

Микросхема MK41T56 (SGS-Thomson) – это маломощный хронометр с объемом памяти 512 бит, которые организованы в 64 байтовых слова. Первые восемь байт используются для хранения значения времени и даты. Устройство поддерживает работу с шиной I^2C в качестве ведомого устройства. Микросхема постоянно следит за напряжением питания: если оно падает ниже определенного уровня, передача данных блокируется, что предотвращает запись ошибочной информации. При использовании литиевой батареи на 3 В, емкостью 30 мА·ч, срок хранения данных превышает 10 лет.

Назначение выводов и внутренняя блок-схема приведены на рис. 7.15. Контакты 8 и 4 соединены с положительным и отрицательным проводами источника питания на +5 В

Рис. 7.15. Назначение выводов и внутренняя блок-схема MK41T56

Устройство потребляет ток 3 мА в активном режиме и 1 мА в режиме ожидания (SDA = SCL = 1). Входы OSC0 и OSC1 соединены с кварцем 32,768 МГц. SCL – это линия синхронизации, а SDA – двунаправленная линия данных шины I²C. FT/OUT – выход проверки частоты. После записи во внутренний регистр микросхемы соответствующего управляющего слова на выходе появляется прямоугольный сигнал частотой 512 Гц. Этот выход может использоваться как программно управляемый.

Функции 64 байт ячеек памяти таковы:

адрес=0	регистр секунд (биты 0–6, 00–59, двоично-десятичный формат)
адрес=1	регистр минут (биты 0–6, 00–59, двоично-десятичный формат)
адрес=2	регистр часов (биты 0–5, 00–23, двоично-десятичный формат)
адрес=3	регистр дней (биты 0–2, 01–07, двоично-десятичный формат)
адрес=4	регистр чисел (биты 0–5, 01–31, двоично-десятичный формат)
адрес=5	регистр месяцев (биты 0–4, 01–12, двоично-десятичный формат)
адрес=6	регистр лет (биты 0–7, 00–99, двоично-десятичный формат)
адрес=7	регистр управления (см. далее)
адрес=8–63	ОЗУ

Функции битов регистра управления определены следующим образом:

бит 7	управление выходом, 0 или 1
бит 6	бит проверки частоты (если 1, проверка частоты)
бит 5	знаковый бит
биты 4–0	не используются

Данные можно считать или записать в микросхему по шине I²C. Операция записи устанавливает значения времени и даты. Операция считывания извлекает из нее данные. После генерации условия «СТАРТ» в память микросхемы заносятся 8 бит данных из ведущего передатчика. Адрес ведомой микросхемы имеет следующий формат (старший разряд слева): 1, 1, 0, 1, 0, 0, 0, $R/\overline{W}$. Эти биты составляют постоянный адрес хронометра на шине. Бит $R/\overline{W}$ определяет тип операции – запись ($R/\overline{W}$ = 0) или чтение ($R/\overline{W}$ = 1). Когда адрес ведомой микросхемы передан, в нее посылается адрес, определяющий конкретную ячейку памяти. Он записывается в регистр адреса и принимает значения от 0 до 64. Затем в режиме записи передаются 8 бит данных, которые и помещаются в указанную ячейку памяти. После этого в режиме чтения снова генерируется условие «СТАРТ», потом передаются биты адреса ведомой микросхемы, где бит $R/\overline{W}$ установлен в 1. И наконец, считываются данные, хранящиеся в указанной ячейке памяти. Временные диаграммы работы ИС представлены на рис. 7.16.

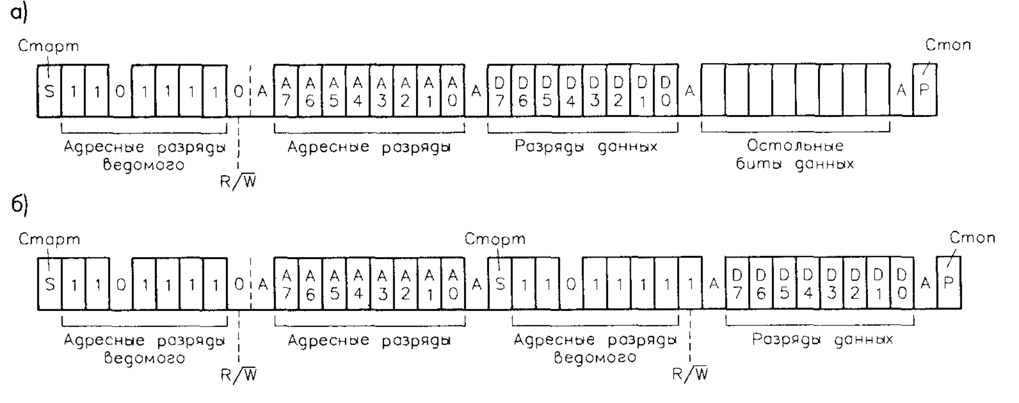


Рис. 7.16. Временные диаграммы работы МК41Т56: а – запись; б – чтение

Схема соединения МК41Т56 с экспериментальной платой параллельного порта показана на рис. 7.17. Контакт D1 управляет линией SDA, данные с которой считываются при помощи контакта S1. Контакт C1 управляет линией SCL.

Текст программы 4156.PAS на TP6

```
Program IIC_timer,
(*Программа управления хронометром MK41T56 *)
(*Компьютер выступает в качестве ведущего устройства, другие микросхемы ведомые *)
(*Соединение с экспериментальной платой параллельного порта:
  D1 = SDA, C1 = SCK, S1 = SDA *)
```

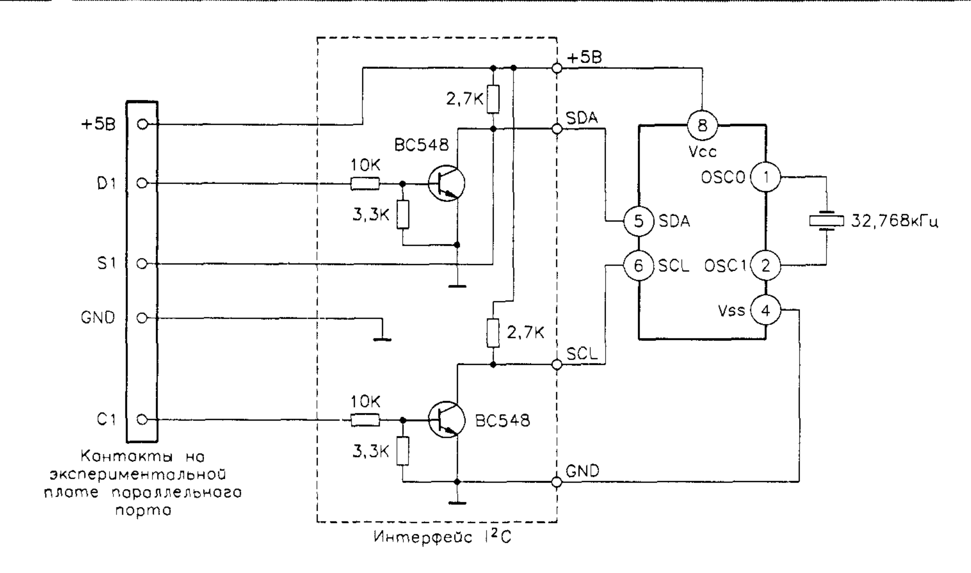


Рис. 7.17. Экспериментальная схема MK41T56

```
(*Условия шины I²C:
  SCL=1, SDA=1:          шина не занята
  SCL=1, SDA=1 - 0:      условие "СТАРТ"
  SCL=1, SDA=0 - 1:      условие "СТОП"
  допустимые данные:     данные постоянны, когда SCL=1
  изменение данных:      когда SCL=0 *)
uses
  crt,dos,

{$I c:\ioexp\tplib1\pas}

var
  i,ij:integer,
  second,minute,hour,day,date,month,year,control_word:byte;

(*TP6 программная библиотека I²C № 1, преобразование в двоично-десятичный формат *)
Function BCD(data.byte):byte,
(*Преобразование двоичного кода в двоично-десятичный.*)
begin
  BCD:=round((data div 10)*16+10*frac(data/10));
end;

(*TP6 программная библиотека I²C № 2, запись данных в линию SDA.*)
Procedure SDA(data.byte);
(*Помещение данных в линию SDA.*)
begin
  write_data_port(P_address,1-data),
end;

(*TP6 программная библиотека I²C № 3, управление линией SCL.*)
Procedure SCL(data:byte);
(*Помещение данных в линию SCL *)
```

```
begin
  write_control_port(P_address,1-data);
end;

(*TP6 программная библиотека I²C № 4, инициализация шины I²C.*)
Procedure INIT;
(*Генерация условия инициализации, SDA=SCL=1.*)
begin
  SDA(1);
  SCL(1);
  delay(100);
end;

(*TP6 программная библиотека I²C № 5, генерация условия "СТАРТ".*)
Procedure START;
(*Генерация условия "СТАРТ".*)
begin
  SDA(1); SCL(1); SDA(0); SCL(0);
end;

(*TP6 программная библиотека I²C № 6, генерация тактовых импульсов для подтверждения.*)
Procedure ACK;
(*Генерация подтверждения и соответствующих тактовых импульсов.*)
begin
  SCL(1); SCL(0);
end;

(*TP6 программная библиотека I²C № 7, передача данных в шину I²C *)
Procedure TRANSMIT(data:byte);
(*Передача данных через шину.*)
var
  i:byte;
begin
  for i:=8 downto 1 do
     begin
        SDA(round(data and bit_weight(i)/bit_weight(i))); (*Помещение данных в линию SDA,
                                                            когда SCL=0.*)
        SCL(1);      (*SCL=0 - 1.*)
        SCL(0);      (*SCL=1 - 0.*)
     end;
  SDA(1);      (*SDA становится равным 1 *)
  ACK;         (*Генерация тактовых импульсов подтверждения.*)
end;

(*TP6 программная библиотека I²C № 8, прием данных от шины I²C.*)
Function receive(stop_flag:boolean);byte;
(*Прием данных через шину.*)
var
  i,dummy:byte;
begin
  dummy:=0;
  for i:=8 downto 1 do
     begin
        SCL(1);         (*SCL становится равным 1.*)
        delay(1);
```

```
            dummy:=dummy+(read_status_port(P_address) and 1)*bit_weight(i);
            (*Считывание данных с линии SDA.*)
            delay(1);
            SCL(0);       (*SCL=1 - 0.*)
        end;
    if stop_flag then
        begin
            SDA(1);       (*Если приняты последние данные, то SDA=1.*)
            ACK;
        end
            else
        begin
            SDA(0);       (*Если принятые данные еще не последние, то SDA=0 для подтверждения.*)
            ACK;          (*Генерация тактовых импульсов подтверждения.*)
            SDA(1);
        end;
    receive:=dummy;
end;

(*TP6 программная библиотека I2C № 9, генерация условия "СТОП".*)
Procedure STOP;
(*Генерация условия "СТОП".*)
begin
    SDA(0);     (*SDA=0.*)
    SCL(1);     (*SCL=1.*)
    SDA(1);     (*SDA=0 - 1.*)
end;

procedure init_time;
(*Инициализация времени.*)
begin
    write('Input initial second:');          read(second);
    write('Input initial minute:');          read(minute);
    write('Input initial hour:');            read(hour);
    write('Input initial day:');             read(day);
    write('Input initial date:');            read(date);
    write('Input initial month:');           read(month);
    write('Input initial year:');            read(year);
    write('Input initial control word:');    read(control_word);
end;

procedure init_41T56(second,minute,hour,day,date,month,year,control_word: byte);
(*Установка значения секунд, минут, часов, дня недели, числа, месяца и года.*)
begin
    start;                      (*Генерация условия "СТАРТ".*)
    transmit(208);              (*Передача подчиненного адреса, R/W=0.*)
    transmit(0);                (*Смещение указателя на адрес 9.*)
    transmit(BCD(second));      (*Передача секунд.*)
    transmit(BCD(minute));      (*Передача минут.*)
    transmit(BCD(hour));        (*.....*)
    transmit(BCD(day));
    transmit(BCD(date));
    transmit(BCD(month));
    transmit(BCD(year));
    transmit(BCD(control_word));
end;
```

```
function MK41T56(x:byte):real;
var
  data:array[1..500] of byte;
begin
  start;                    (*Генерация условия "СТАРТ".*)
  transmit(208);            (*Передача подчиненного адреса, R/W=0.*)
  transmit(0);              (*Смещение указателя на адрес 0 *)
  start;                    (*Новая генерация условия "СТАРТ" для чтения.*)
  transmit(208+1);          (*Передача подчиненного адреса 208+1 D, R/W=1.*)
  for i:=1 to 6 do data[i]:=receive(false);     (*Считывание данных с подтверждением.*)
  data[7]:=receive(true);   (*Считывание данных без подтверждения.*)
                            (*При чтении адрес регистра увеличивается автоматически.*)
  (*Преобразование данных в формат времени.*)
  if x=1 then MK41T56:=data[1] and (64+32+16)/16*10+data[1] and (8+4+2+1);
  if x=2 then MK41T56:=data[2] and (64+32+16)/16*10+data[2] and (8+4+2+1);
  if x=3 then MK41T56:=data[3] and (32+16)/16*10+data[3] and (8+4+2+1);
  if x=4 then MK41T56:=data[4] and (4+2+1);
  if x=5 then MK41T56:=data[5] and (32+16)/16*10+data[5] and (8+4+2+1);
  if x=6 then MK41T56:=data[6] and (32+16)/16*10+data[6] and (8+4+2+1);
  if x=7 then MK41T56:=data[7] and (128+64+32+16)/16*10+data[7] and (8+4+2+1);
end;

procedure show_time;
begin
  writeln('    Inter IC bus 41T456 Timer');
  writeln;
  write('Input initial second:'); writeln(MK41T56(1):5:0);
  write('Input initial minute:'); writeln(MK41T56(2):5:0);
  write('Input initial hour:'); writeln(MK41T56(3):5:0);
  write('Input initial day:'); writeln(MK41T56(4):5:0);
  write('Input initial date:'); writeln(MK41T56(5):5:0);
  write('Input initial month:'); writeln(MK41T56(6):5:0);
  write('Input initial year:'); writeln(MK41T56(7):5:0);
end;

begin
  centronic_address;
  init_time;
  init;               (*Инициализация линий SCL и SDA.*)
  init_41T56(second,minute,hour,day,date,month,year,control_word);
  repeat
      clrscr;
      show_time;
      delay(10000);
  until keypressed;
end.
```

7.5. Генераторы сигналов с цифровым управлением

Генераторы сигналов с цифровым управлением позволяют формировать цифровые и аналоговые сигналы с частотой, устанавливаемой цифровой схемой управления. Микросхемы 8253/8254 – это интегральные таймеры/счетчики, которые широко используются для генерации цифровых сигналов. Микросхемы HSP45102 и ML2036 – программируемые генераторы синусоидальных колебаний.

7.5.1. Программируемый таймер/счетчик 8254

Микросхема (рис. 7.18) снабжена тремя 16-разрядными программируемыми счетчиками, каждый из которых имеет по три линии ввода/вывода: CLK, GATE и OUTPUT. На вход CLK подается тактовый сигнал частотой от 0 до 8 МГц. Вход GATE управляет запуском и остановкой счетчика. Если на этот вход поступает сигнал высокого уровня, то счетчик включается. На выходе OUTPUT появляется выходной сигнал счетчика. Режимы работы микросхемы задаются с помощью восьмиразрядной двунаправленной шины данных и пяти линий управления (A0, A1, $\overline{WR}$, $\overline{RD}$ и $\overline{CS}$). Данные записываются в четыре внутренних регистра: регистр

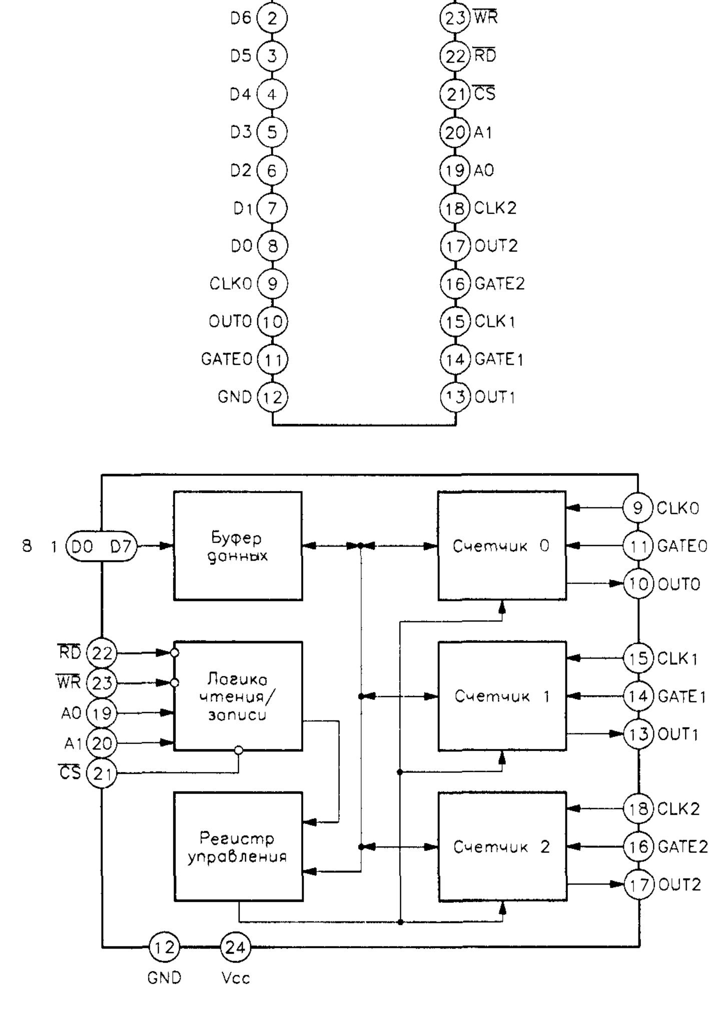


Рис. 7.18. Назначение выводов и внутренняя блок-схема 8253/8254

управления и три регистра счетчиков. Состояние входов A0 и A1 для выбора конкретного регистра можно охарактеризовать следующим образом:

- A0 = 0, A1 = 0 – выбор регистра счетчика 1;
- A0 = 1, A1 = 0 – выбор регистра счетчика 2;
- A0 = 0, A1 = 1 – выбор регистра счетчика 3;
- A0 = 1, A1 = 1 – выбор регистра управления.

Перед использованием счетчики необходимо настроить посредством записи управляющего слова в регистр управления. Функции битов управляющего слова показаны ниже. Биты 7 (SC1) и 6 (SC0) выбирают настраиваемый счетчик:

бит 7 = 0, бит 6 = 0	настройка счетчика 0
бит 7 = 0, бит 6 = 1	настройка счетчика 1
бит 7 = 1, бит 6 = 0	настройка счетчика 2
бит 7 = 1, бит 6 = 1	команда считывания

Биты 5 (RW1) и 4 (RW0) управляют форматом чтения и записи каждого регистра счетчика:

бит 5 = 0, бит 4 = 0	команда загрузки счетчика
бит 5 = 0, бит 4 = 1	чтение/запись только младшего байта
бит 5 = 1, бит 4 = 0	чтение/запись только старшего байта
бит 5 = 1, бит 4 = 1	чтение/запись сначала младшего, затем старшего байта

Биты 3 (M2), 2 (M1) и 1 (M0) управляют выходными режимами. Всего существует шесть выходных режимов:

бит 3 = 0, бит 2 = 0, бит 1 = 0	режим 0 – генерация фронта
бит 3 = 0, бит 2 = 0, бит 1 = 1	режим 1 – одновибратор
бит 3 = x, бит 2 = 1, бит 1 = 0	режим 2 – генератор с изменяемой скважностью
бит 3 = x, бит 2 = 1, бит 1 = 1	режим 3 – генератор прямоугольных импульсов
бит 3 = 1, бит 2 = 0, бит 1 = 0	режим 4 – генератор программного строб-импульса
бит 3 = 1, бит 2 = 0, бит 1 = 1	режим 5 – генератор аппаратного строб-импульса

Бит 0 настраивает формат данных в регистрах счетчиков:

0 – 16-разрядный двоичный счетчик
1 – 4-разрядный двоично-десятичный счетчик

Чтобы записать данные в микросхему, необходимо подать информацию на входы D0 – D7 и A0 и A1 для выбора регистра. Затем по отрицательному импульсу на входе $\overline{WR}$ данные загружаются в выбранный регистр. Во время операции необходимо, чтобы на входе $\overline{CS}$ был низкий уровень сигнала, иначе микросхема будет заблокирована.

Каждый счетчик 8254 имеет шесть выходных режимов, работа которых поясняется на рис. 7.19.

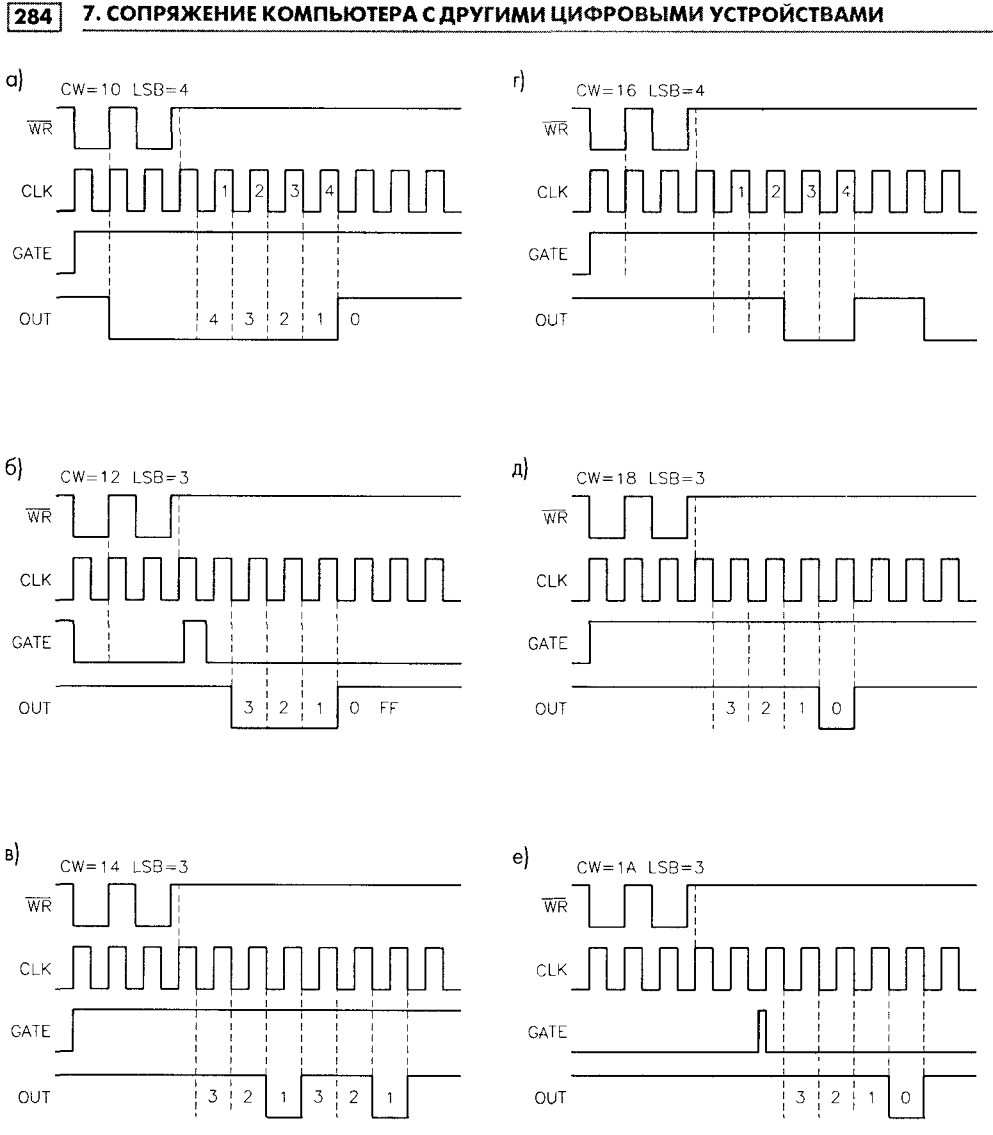


Рис. 7.19. Выходные режимы 8254 а – генератора фронта, б – одновибратора, в – генератора импульсов с изменяемой скважностью, г – генератора прямоугольных импульсов, д – генератора программного строб-импульса; е – генератора аппаратно-программного строб-импульса

Режим генератора фронта (режим 0, рис. 7.19а) используется для генерации прерывания на выходе по истечении определенного промежутка времени. После записи в регистр управления управляющего слова числа 10 (десятичное) на выходе счетчика 0 устанавливается низкий уровень. Когда количество тактовых импульсов на единицу превысит значение, записанное в счетчик, на его выходе появляется сигнал высокого уровня.

Режим одновибратора (режим 1, рис. 7.19б) применяется для генерации отрицательного импульса. Длительность периода времени, когда на выходе счетчика удерживается низкий уровень, зависит от частоты тактовых импульсов и числа, записанного в счетчик.

Режим генератора импульсов с изменяемой скважностью (режим 2, рис. 7.19в). Если в счетчик записано число N, то сигнал низкого уровня на его выходе появится через N тактов. На следующем такте сигнал на выходе счетчика снова станет единицей. Таким образом, счетчик работает как генератор импульсов с частотой, равной тактовой частоте, деленной на N.

Режим генератора прямоугольных импульсов (режим 3, рис. 7.19г). Если в счетчик записано четное число, то на выходе будет сигнал прямоугольной формы со скважностью 50%. Частота сигнала равна тактовой частоте, деленной на число, которое записано в счетчик. Если записанное число нечетное, то длительность единицы будет на один тактовый интервал больше, чем длительность нуля.

Режим генератора программного строб-импульса (режим 4, рис. 7 19д). В этом режиме значение счетчика автоматически уменьшается сразу после записи в него начального значения, причем скорость уменьшения равна тактовой частоте. Как только значение счетчика достигает нуля, он вырабатывает отрицательный импульс длительностью, равной длительности тактового импульса. Таким образом, импульс появляется через N+1 тактов после записи в счетчик числа N.

Режим аппаратно генерируемого строб-импульса (режим 5, рис. 7.19е) аналогичен режиму 4, однако начало обратного отсчета инициируется положительным импульсом на входе GATE. По положительному фронту импульса на входе GATE начинается процесс отсчета. Со следующим тактовым импульсом значение счетчика уменьшается, а когда оно достигает нуля, на выход OUT поступает отрицательный импульс длительностью, равной длительности одного тактового импульса. Таким образом, на выходе OUT импульс появляется через N+1 тактов после подачи на вход GATE сигнала высокого уровня.

На рис. 7.20 изображена схема на основе программируемого таймера/счетчика 8254, подключенного к экспериментальной плате параллельного порта. Контакты D1 и D2 на плате предназначены для последовательного ввода данных в регистр сдвига 74LS164, с восьмиразрядного параллельного выхода которого данные подаются в микросхему 8254. Контакты D5, D6, D7 и D8 соединены с входами $\overline{WR}$, A1, A0 и $\overline{CS}$.

Для опытов используется счетчик 2. На вход CLK2 подается тактовый сигнал, на вход GATE 2 – сигнал высокого уровня для разблокирования счетчика. Программа на языке TP6 переводит счетчик в режим генератора прямоугольного напряжения. Частота сигнала устанавливается программно.

Текст программы 8254.PAS

```
Program mini_sig_generator;
(*Программа управления таймером/счетчиком 8254 *)
(* Соединение с экспериментальной платой параллельного порта
   D1=данные, D2=такты для 74LS164,
   D5=WR, D6=A1, D7=A0 и D8=CS для 8253 *)
```

Рис. 7.20. Схема на базе программируемого таймера/счетчика 8254

```
uses
  dos,crt,

{$I c \ioexp\tplib1 pas}

const
  base_frequency=2457600,
(*Частота тактового сигнала 8254 *)

var
  command byte,
  output_frequency longint,

procedure setbit(bitnumber, bitvalue byte),
(*Установка значения определенного бита команды  bitnumber=1-8  bitvalue=0 или 1 *)
begin
  if bitvalue=1 then command =command or bit_weight(bitnumber),
  if bitvalue=0 then command =command and (255-bit_weight(bitnumber)),
end,

procedure initialization,
(*Установка всех линий в 1 *)
```

```
begin
  command:=127;
  write_data_port(P_address,command);  (*Ввод команды в порт данных.*)
  output_frequency:=1000;
end;

procedure loaddata(address,data:byte);
(* Загрузка данных в регистр сдвига 74LS164 и запись в 8254,
  address выбирает счетчик (0-2), регистр управления data определяет данные для записи
  в регистры.
  Во время загрузки   (1) в D0 загружаются данные sw[i]
                      (2) на D1 (CLOCK) подается перепад 0-1-0
                      (4) D7 (CS) = 0 для разблокирования ИС
                      (5) в D6 и D5 (A0 и A1) должен быть загружен необходимый адрес
                      (6) на D4 (WR) подается перепад 1-0-1, после загрузки
                      D4 все время должно быть равно 1.*)
var
  d:array[1..8] of byte;
  i,A0,A1:byte;
begin
  if address=0 then begin A0:=0; A1:=0; end;
  if address=1 then begin A0:=1; A1:=0; end;
  if address=2 then begin A0:=0; A1:=1; end;
  if address=3 then begin A0:=1; A1:=1; end;

  (*Определение битов данных для последовательной передачи в 74LS164.*)
  for i:=8 downto 1 do
     begin
        d[i]:=0;
        if data>=bit_weight(i) then
           begin
              data:=data-bit_weight(i);
              d[i]:=i;
           end;
     end;

  (*Загрузка данных в регистры 74LS164.*)
  for i:=1 to 8 do
     begin
        setbit(1,d[i]); write_data_port(P_address,command);
        setbit(2,0);    write_data_port(P_address,command);
        setbit(2,1);    write_data_port(P_address,command);
        setbit(2,0);    write_data_port(P_address,command);
     end;

  (*Занесение данных в ИС 8254.*)
  setbit(7,A0);
  setbit(6,A1);   write_data_port(P_address,command);
  setbit(5,0);    write_data_port(P_address,command);
  setbit(5,1);    write_data_port(P_address,command);
end;

procedure signal_generator(base_frequency,frequency:longint);
(*Настройка счетчика 2 в режим 3 - режим генератора сигналов.*)
var
  divisor:longint;
  high_byte_0,low_byte_0:byte;
```

```
begin
  divisor:=round(base_frequency/frequency);
  high_byte_0:=divisor div 256;
  low_byte_0:=divisor mod 256;
  if divisor>65000 then writeln('Error in delay time');
  loaddata(3,$b6);         (*Загрузка управляющего слова.*)
  loaddata(2,low_byte_0);  (*Загрузка младшего байта.*)
  loaddata(2,high_byte_0); (*Загрузка старшего байта.*)
end;

procedure test_8253;
begin
  write('Input output frequency [Hz] (0 to quit):');
  readln(output_frequency);
  signal_generator(base_frequency,output_frequency);
end;

(*Главная программа.*)
begin
  centronic_address;
  initialization;
  repeat
     if output_frequency>0 then test_8253
  until output_frequency=0;
end.
```

7.5.2. Генератор с числовым программным управлением HSP45102

Микросхема HSP45102 (Harris Semiconductor, RS284-977) – это генератор с числовым программным управлением, который последовательно выдает 12-разрядный двоичный код, представляющий собой значения синусоидальной функции за один период. Частота и фаза колебания задаются программно. Частота сигнала определяется одним из двух предустановленных 32-разрядных слов, которые объединены в один 64-разрядный регистр. Слово, указывающее выходную частоту, выбирается с помощью входа управления $SEL_L/\overline{M}$. Фаза сигнала зависит от состояния входов P0 и P1, которые устанавливают начальный сдвиг фазы 0°, 90°, 180° и 270°.

Назначение выводов и внутренняя блок-схема приведены на рис. 7.21. Входы Vcc (контакты 8 и 22) и GND (контакты 7, 15 и 21) соединены с положительным и нулевым проводами источника питания. Напряжение питания +5 В. Потребляемый ток в активном режиме равен 99 мА, а в режиме ожидания – 500 мкА. CLK (контакт 16) – это вход тактового сигнала. Максимальная тактовая частота 33 МГц.

SCLK (контакт 14), SD (контакт 13), $MSB/\overline{LSB}$ (контакт 11) и $\overline{SFTEN}$ (контакт 10) – входы секции управления частотой. SCLK и SD – тактовый вход и вход последовательных данных. Данные на входе SD сдвигаются во внутренние регистры микросхемы по положительному фронту импульсов SCLK. $\overline{SFTEN}$ – это вход разрешения сдвига. Для разрешения сдвига данных на него необходимо подать низкий уровень. Если $MSB/\overline{LSB}$ = 1, то первый сдвигаемый бит интерпретируется как старший, если 0 – как младший. Значение генерируемого гармонического колебания можно вычислить по формуле:

Выходная частота (Гц) = $N \times Fclk/2^{32}$,

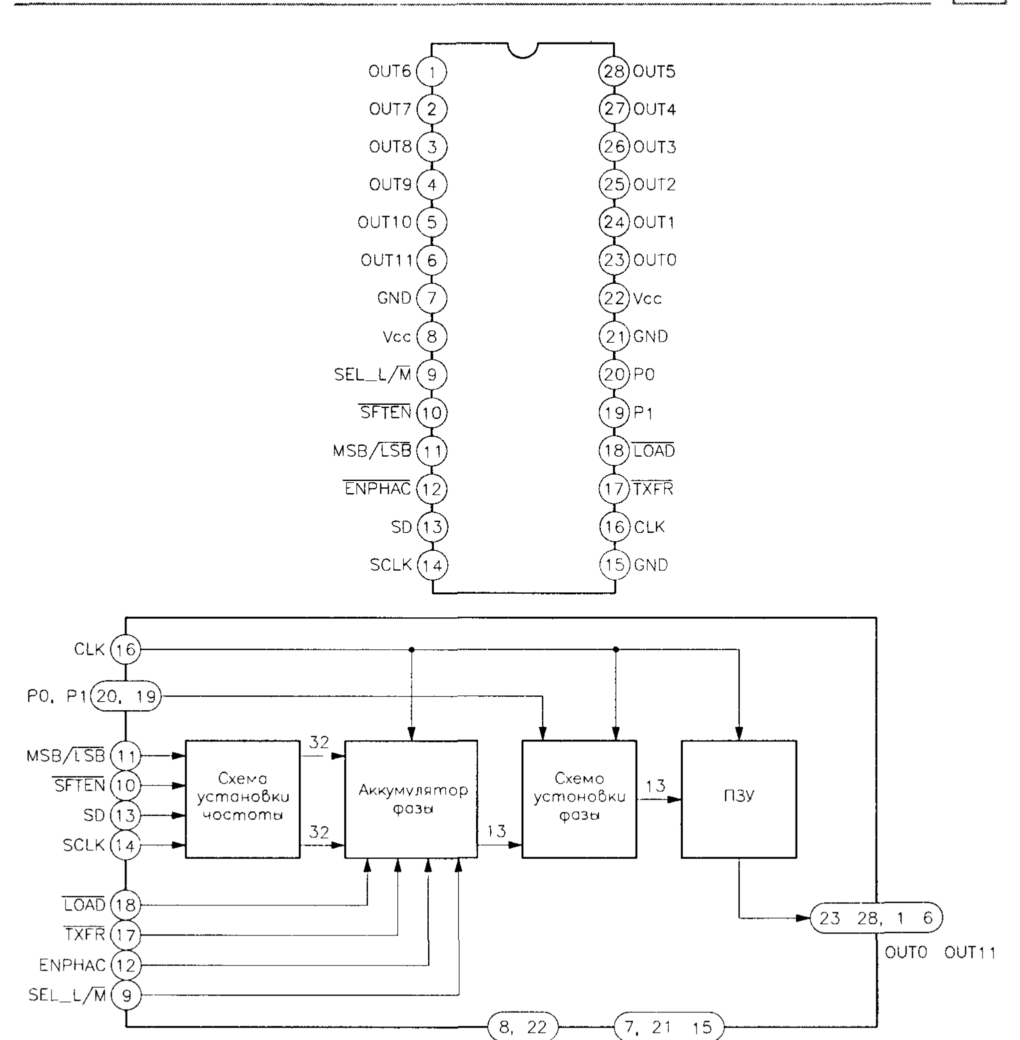


Рис. 7.21. Назначение выводов и внутренняя блок-схема генератора HSP45102

где N – число, записанное в выбранное слово установки частоты, а Fclk – частота тактового сигнала.

$\overline{\text{LOAD}}$ (контакт 18), $\overline{\text{TXFR}}$ (контакт 17), $\overline{\text{ENPHAC}}$ (контакт 12) и SEL_L/$\overline{\text{M}}$ (контакт 9) – это управляющие входы аккумулятора фазы. $\overline{\text{ENPHAC}}$ открывает аккумулятор фазы, SEL_L/$\overline{\text{M}}$ устанавливает слово выбора частоты. Если на него подан сигнал высокого уровня, то указываются младшие 32 разряда 64-разрядного регистра частоты, а если низкого – старшие.

Если на входе $\overline{\text{TXFR}}$ низкий уровень, то слово выбора частоты, определенное значением SEL_L/$\overline{\text{M}}$, передается из регистра частоты во входной регистр аккумулятора фазы. Входы P0 (контакт 20) и P1 (контакт 19) – это входы установки

фазы, с помощью которых можно задать сдвиг фазы 0°, 90°, 180° и 270°. Микросхема имеет 12-разрядную выходную шину данных (контакты 1–6, 23–28). Значения данных находятся в диапазоне 000h–FFFh, нулевой уровень гармонического колебания соответствует коду 800h. Для преобразования 12 бит данных в аналоговый сигнал необходимо использовать ЦАП.

Схема на базе генератора HSP45102 изображена на рис. 7.22. Контакты D1 и C1 на плате соединены с выводами SD и SCLK микросхемы HSP45102, вывод CLK (контакт 16) – с выходом тактового генератора. Восемь старших разрядов выходной шины данных микросхемы (выходы OUT 4 – OUT 11) подключены к восьмиразрядному ЦАП ZN428E. Остальные четыре выхода не используются. Управляющие входы соединены либо с «землей», либо с проводом +5 В. Выходной сигнал ЦАП (гармоническое колебание) можно наблюдать на экране осциллографа.

Текст программы 45102.PAS на TP6

```
Program HSP45102,
(*Программа управления для генератора HSP45102 *)
```

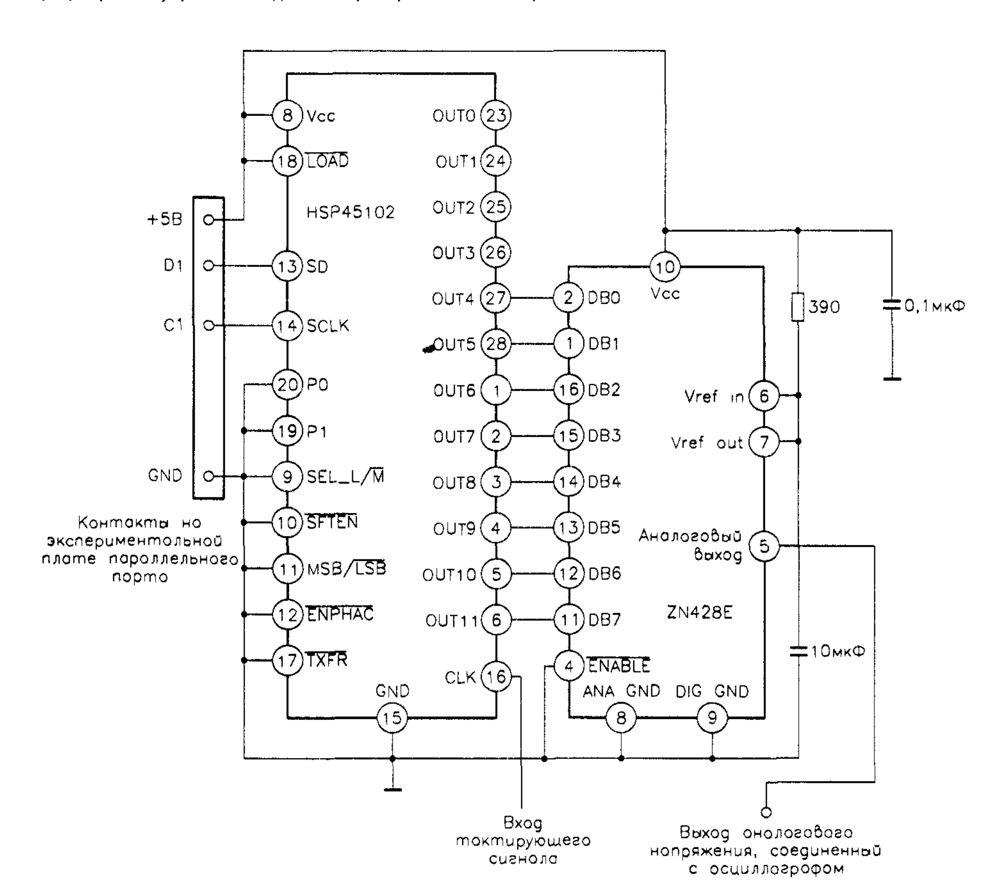


Рис. 7.22. Схема с использованием генератора HSP45102

```
(* Соединение с экспериментальной платой параллельного порта:
  D1 соединен с SD, C1.- с SCLK.*)
uses
  crt,dos;
{$I c:\ioexp\tplib1.pas}
var
  output_frequency:real;
procedure load_frequency(frequency:real);
(*Загрузка значения частоты в регистр частоты генератора.*)
(*Значения частоты приведены в герцах.*)
var
  i,j,k:byte;
  bytex:array[1..4] of byte;
  n:longint;
begin
  n:=round(frequency*256*256*256*256/2.457e6);
  (*Преобразование n в четыре восьмиразрядных байта bytex[1] - bytex[4].*)
  bytex[4]:=round(n/256/256/256);
  n:=n-bytex[4]*256*256*256;
  bytex[3]:=round(n/256/256);
  n:=n-bytex[3]*256*256;
  bytex[2]:=round(n/256);
  n:=n-bytex[2]*256;
  bytex[1]:=round(n);
  (*Загрузка первых 32 бит данных.*)
  for i:=1 to 4 do
      begin
          for j:=1 to 8 do
              begin
                  write_data_port(P_address,round(bytex[i] and bit_weight(j)/bit_weight(j)));
                  write_control_port(P_address,0);
                  write_control_port(P_address,1);
                  write_control_port(P_address,0);
              end;
      end;
  (*Загрузка вторых 32 бит данных.*)
  for i:=1 to 4 do
      begin
          for j:=1 to 8 do
              begin
                write_data_port(P_address,round(bytex[i] and bit_weight(j)/bit_weight(j)));
                write_control_port(P_address,0);
                write_control_port(P_address,1);
                write_control_port(P_address,0);
              end;
      end;
end;
procedure test_45102;
begin
  write('Input output frequency [Hz] (0 to quit):');
  readln(output_frequency);
  load_frequency(output_frequency);
end;
```

```
begin
  centronic_address;
  repeat
     test_45102;
  until output_frequency=0;
end
```

7.5.3. Программируемый генератор синусоидальных колебаний ML2036

Микросхема ML2036 (Micro Linear) представляет собой генератор синусоидальных колебаний, который формирует гармонический сигнал частотой от 0 до 50 кГц при помощи нескольких дополнительных элементов обвязки. Управляющие данные загружаются в ИС последовательно по положительному фронту тактового импульса. С восьмиразрядного ЦАП аналоговый сигнал поступает на ФНЧ, который нужен для улучшения формы гармонического колебания, подаваемого на выход микросхемы. Генерируемое колебание имеет амплитуду ±Vref или ±0,5Vref в зависимости от настройки управляющих входов. При использовании кварца на 4,1943 МГц генератор может выдавать сигнал частотой от 0,5 Гц до 32,768 кГц. Выходная частота вычисляется по следующей формуле:

Выходная частота (Гц) = частота × N/8388680,

где N – число, записываемое в счетчик микросхемы ML2036.

8. СЕТЕВЫЕ ПРИЛОЖЕНИЯ И УДАЛЕННЫЙ ДОСТУП

Глава посвящена вопросам дистанционного управления компьютером, а также сетевым приложениям. Приводится описание электронных устройств, осуществляющих обмен цифровыми данными с помощью средств радиосвязи.

8.1. Телекоммуникационные схемы

Микросхема PCD3311C (Philips Semiconductors, RS273-545) генерирует двухтональные частоты в формате DTMF или несущие частоты для модемов (рис. 8.1). Она питается от источника +5 В и потребляет ток 0,9 мА. Ток в режиме ожидания составляет всего 3 мкА.

Встроенный тактовый генератор с кварцевой стабилизацией частоты требует наличия внешнего кварца на 3,58 МГц, который подключается к выводам OSC1 и OSC0. Микросхема соединяется с компьютером через параллельный порт или шину I^2C. Если на вход MODE (режим) подать сигнал высокого уровня, то ИС начнет работать через параллельную шину; если же оставить вход неподключенным или подать низкий уровень, будет использоваться шина I^2C. Выходной сигнал с выхода TONE фильтруется посредством встроенного фильтра с управляемыми конденсаторами и активного RC-фильтра нижних частот. Благодаря встроенному источнику опорного напряжения выходной сигнал имеет уровни напряжения от 150 до 190 мВ (среднеквадратическое значение).

В режиме параллельного обмена данными PCD3311C настраивается с помощью управляющего слова, записываемого через входы D0 – D5. Данные на указанные входы должны подаваться до появления положительного фронта на входе STROBE. По отрицательному фронту этого импульса информация заносится в микросхему, а на выходе TONE формируется выходной сигнал. Сигнал на выходе остается неизменным до прихода следующего отрицательного фронта на вход STROBE. Временные диаграммы работы микросхемы показаны на рис. 8.2.

Рис. 8.1. Назначение выводов и внутренняя блок-схема PCD3311C

Входы D5 и D4 выбирают режимы работы, а D3 – D0 – комбинацию частот (DTMF или несущие частоты для модемов).

Микросхему PCD3311C можно подключить к экспериментальной плате параллельного порта для получения системы набора телефонного номера на базе ПК. При этом выводы D0 – D5 необходимо соединить с контактами D1 – D6, а вход STROBE – с контактом C1.

8.2. Интегральные схемы модемов

Микросхема TCM3105 (Texas Instruments) – это *модем с частотной манипуляцией* (ЧМ), функционирующий в полосе стандартного телефонного канала (рис. 8.3). Модем работает в дуплексном режиме. Передатчик представляет собой программируемый синтезатор частот, который формирует две частоты, соответствующие

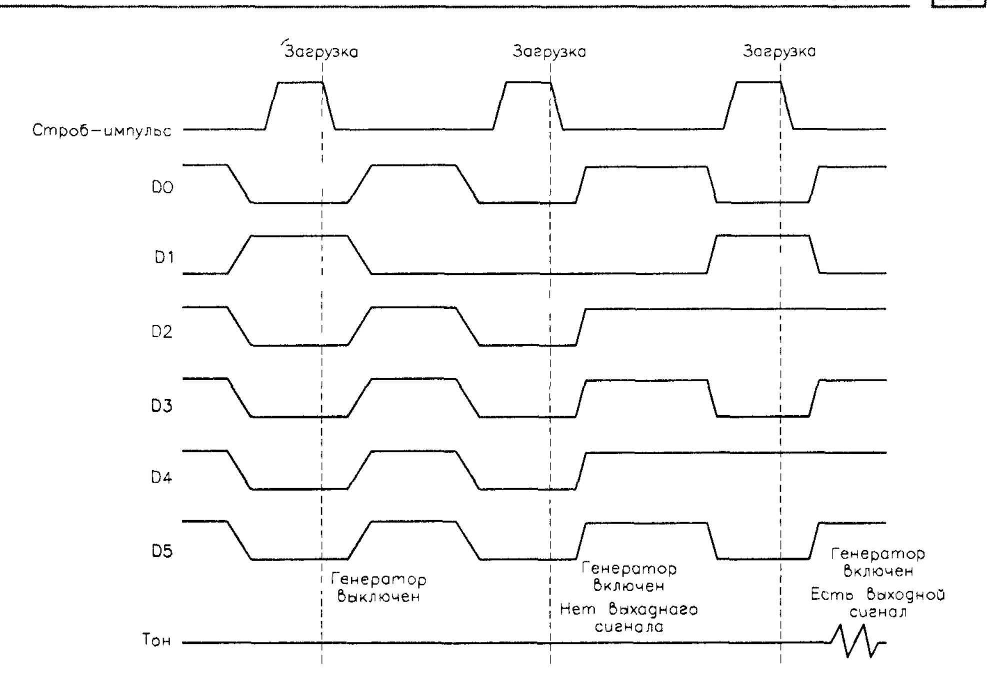


Рис. 8.2. Временные диаграммы режима параллельного обмена данными

0 и 1 двоичного кода. Приемник демодулирует входной аналоговый сигнал и выдает на выход либо 0, либо 1 в зависимости от входной частоты.

Напряжение источника питания равно +5 В, потребляемый ток – 5,5 мА. К выводам OSC1 и OSC2 подсоединен кварц на 4,4336 МГц. С помощью входов TXR1 и TXR2 выбирается скорость работы модема, а посредством TRS – используемый протокол. TXD и TXA – это цифровой вход и аналоговый выход передатчика. RXA и RXD – аналоговый вход и цифровой выход приемника. CDT – выход детектора несущей. Низкий уровень на нем означает, что приемник не может принять сигнал. Порог определения несущей регулируется по входу CDL.

Передатчик состоит из ЧМ модулятора, представляющего собой программируемый синтезатор частоты, который генерирует частоты делением тактовой частоты 4,4336 МГц. Коэффициент деления устанавливается входами TRS, TXR1 и TXR2. Если TRS = TXR1 = TXR2 = 0, то частота для передачи единицы равна 1300, а для нуля – 2100 Гц. Приемник демодулирует входной аналоговый сигнал.

Для получения модема микросхема соединяется с экспериментальной платой последовательного порта. Канал связи можно организовать, используя различные физические носители: звуковые или световые волны, а также радиоволны.

8.3. Радиосвязь

В разделе приводится описание интегральных микросхем, которые предназначены для обмена цифровыми данными с помощью радиосвязи.

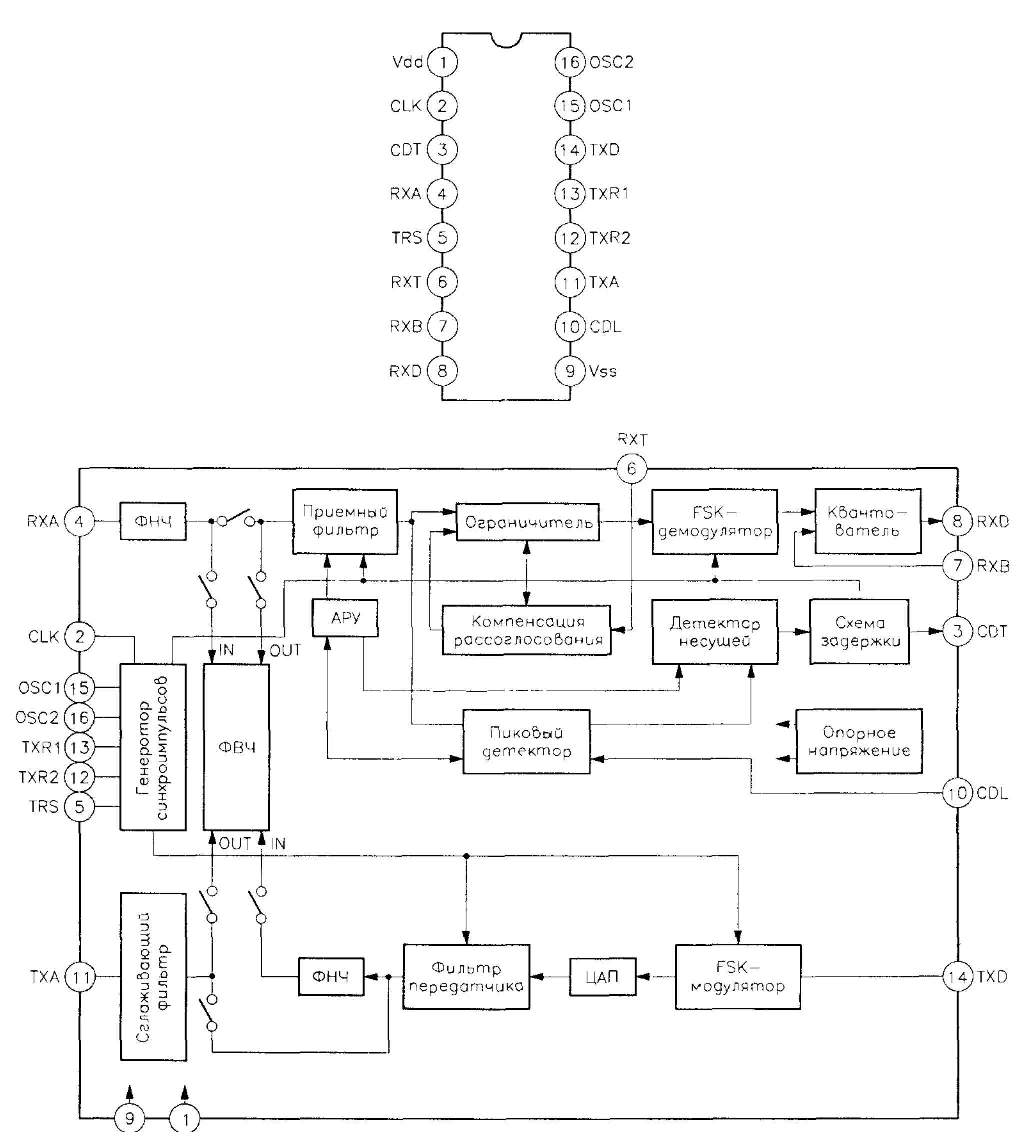


Рис. 8.3. Назначение выводов и внутренняя блок-схема микросхемы TCM3105

8.3.1. FM передатчик и приемник TMX/SILRX

Микросхемы FM радиоприемников и передатчиков работают на частотах 418 и 433 МГц и специально созданы для телеметрии и приложений дистанционного управления. Для передачи на этих частотах необходимо получить разрешение соответствующих ведомств[1].

[1] В Российской Федерации вопросы распределения радиочастотного ресурса находятся в компетенции Государственного комитета по радиочастотам. – *Прим. науч. ред.*

Передатчик TMX

Назначение выводов и внутренняя блок-схема передатчика TMX (Radiometrix, RS740-290) приведены на рис. 8.4. Напряжение питания подается на вход Vcc (контакт 3). Выводы 1 и 4 внутрисхемно соединены и образуют общий провод.

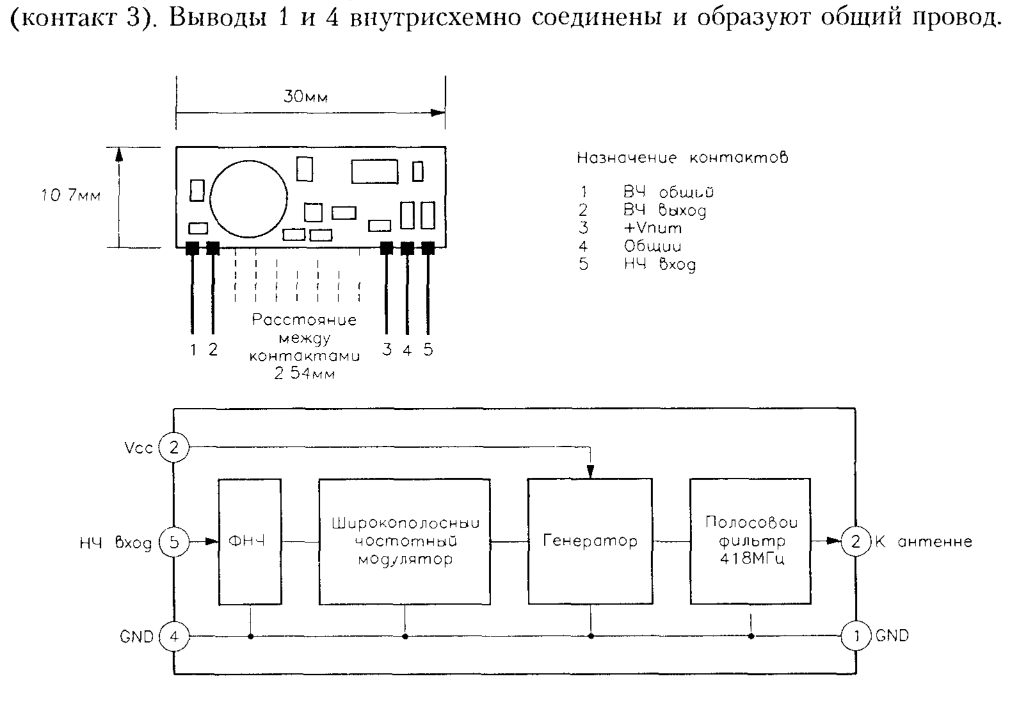


Рис. 8.4. Назначение выводов и внутренняя блок-схема передатчика TMX-418-A

Передатчик функционирует в широком диапазоне напряжения питания (+6...+12 В) и имеет номинальный рабочий ток 6 мА при напряжении 6 В. Рабочие частоты 418 и 433 МГц. DATA IN (контакт 5) – вход модулятора, совместимый с уровнями напряжений логики КМОП при условии, что схема управления имеет такое же напряжение питания. Антенна соединяется с контактом 2. Полученные на входе цифровые данные подаются на RC-фильтр нижних частот, который ограничивает полосу частот модулированного сигнала до 10 кГц. Затем они поступают на частотный модулятор. Модулятор управляется варикапом: изменяя емкость последнего, можно менять частоту генератора. Центральная частота генератора равна 418 МГц и определяется резонатором поверхностных акустических волн.

Промодулированный высокочастотный сигнал подается на контакт 2, который соединен с антенной.

Антенна передатчика бывает трех типов: спиральная, рамочная или штыревая (см. рис. 8.6). *Спиральная антенна* небольшого размера (17 мм длиной и 2,5 мм диаметром). Она имеет высокую добротность и, следовательно, нуждается в настройке для конкретной длины волны. *Рамочная антенна* состоит из рамки в виде

дорожки печатной платы, которая настраивается с помощью переменного конденсатора. *Штыревая антенна* выполнена из провода, стержня, дорожки печатной платы длиной примерно 16,5 см.

Модуль радиопередатчика сертифицирован в Сообществе MPT1340[1], его использование на территории Великобритании разрешено для телеметрии, дистанционного управления и разработки устройств, удовлетворяющих следующим требованиям:

- передающая антенна относится к одному из трех вышеперечисленных типов;
- модуль передатчика соединен непосредственно с антенной без помощи внешних проводов. Увеличение мощности передатчика какими-либо методами не допускается;
- модуль нельзя модифицировать или использовать вне пределов его установки;
- модуль применяется только для передачи данных. Передача голоса или музыки запрещена;
- оборудование, в котором используется модуль, должно иметь контрольную метку, размещенную внутри блока и хорошо видимую. Минимальные размеры метки 10×15 мм, а высота букв и рисунков – не менее 2 мм. Формулировка может быть примерно следующей: «MPT 1340 W.T. LICENCE EXEMPT»;
- настройка подстроечного конденсатора – заводская установка, конечный пользователь ее изменить не может.

Приемник SILRX

Назначение выводов и внутренняя блок-схема приемника SILRX (Radiometrix, RS740-304) представлены на рис. 8.5.

Напряжение питания подается на контакт 5. Антенный вход – контакт 1. Контакты 2 и 4 выступают в качестве заземления. Модуль работает от напряжения питания 4,5–9 В и потребляет ток порядка 13 мА. Рабочие частоты 418 и 433 МГц. Входной радиосигнал, принятый антенной, подается на полосовой фильтр через конденсатор. Предварительный усилитель радиочастоты усиливает сигнал перед тем, как он поступает на первый каскад смесителя. Первый гетеродин настроен на частоту 433,92 МГц для получения первой ПЧ 15,92 МГц. Затем сигнал проходит на второй смеситель, в котором гетеродин настроен на частоту 16 МГц. Вторая ПЧ равна 80 кГц. После этого сигнал усиливается, демодулируется и поступает на выход (контакт 7) через фильтр нижних частот третьего порядка. Приемник имеет выход ПЧ, который подается на контакт 3. Выход приемника совместим с логическими уровнями КМОП. К приемнику может быть подключена любая из антенн (рис. 8.6).

---

[1] В Российской Федерации вопросы, связанные с сертификацией радиопередатчиков, находятся в компетенции Министерства связи. – *Прим. науч. ред.*

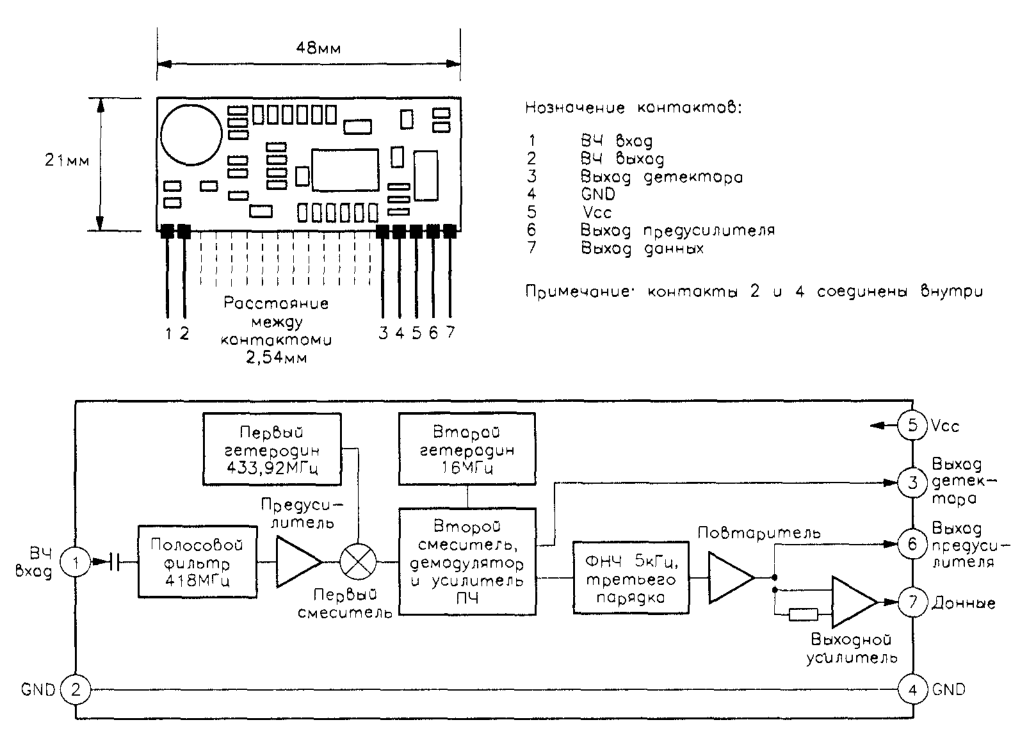


Рис. 8.5. Назначение выводов и внутренняя блок-схема SILRX-418

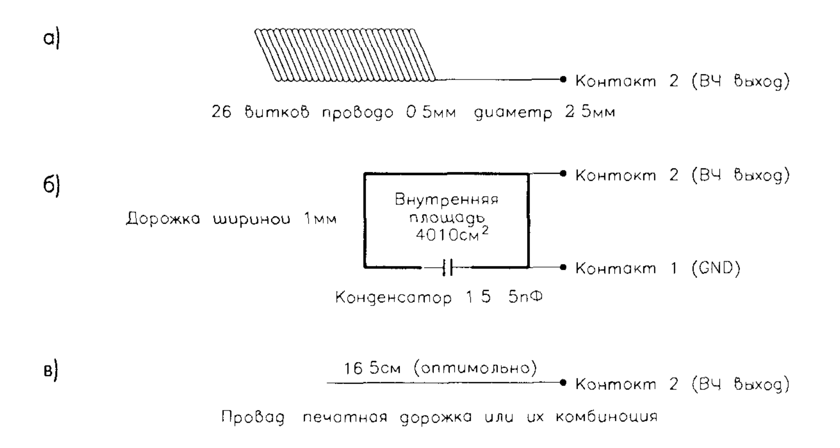
8.3.2. АМ передатчик и приемник AM-TX1/AM-HHR3

Микросхема AM-TX1 (RF Solutions) – это миниатюрный передатчик в гибридном исполнении с амплитудной манипуляцией (рис. 8.7), его можно использовать для передачи данных от любого стандартного КМОП/ТТЛ источника со скоростью 1200 бод. Для управления передатчиком нужно всего две линии. Модуль экономичен, потребляет 2,3 мА. Выход соединен с рамочной или штыревой антенной. Дальность действия более 100 м. Рабочие частоты 418 и 433 МГц. Устройство сертифицировано MPT1340 для применения в телеметрии и связи при условии использования антенн, изображенных на рис. 8.6.

Микросхема AM-HHR3 (RF Solutions) – это компактный модульный радиоприемник, который можно использовать для приема сигнала, посылаемого АМ передатчиком. Расположение выводов приведено на рис. 8.8. Выход совместим с уровнем ТТЛ/КМОП. Напряжение источника питания +5 В, потребляемый ток 2,5 мА.

8.3.3. Эксперименты по передаче данных с помощью радиосвязи

На рис. 8.9 изображена схема, преобразующая 12 бит данных из параллельной формы в последовательную для передачи последовательных данных с помощью



г)

Характеристики антенн	Спиральная	Рамочная	Штыревая
Эффективность	√ √	√	√ √ √
Простота установки	√ √	√	√ √ √
Размер	√ √ √	√ √	√
Стабильность характеристик	√ √	√ √ √	√

Рис. 8.6. Различные типы антенн для радиопередатчиков TXM-418 а – спиральная, б – рамочная, в – штыревая, г – характеристики антенн

11 5мм
+V SIDE
+V
7 5мм
GND
Медный провод диаметром 1 1 5мм
Uвх =+5В от ТТЛ или КМОП (0 4 5В)
330
Максимальная площадь 700мм²
220пФ
+V
AX TX1
1–5пФ
Поверхность земли
ВЧ общий
Перестраиваемая рамочная антенна

Рис. 8.7. Назначение выводов и типовое включение модуля передатчика AM TX1

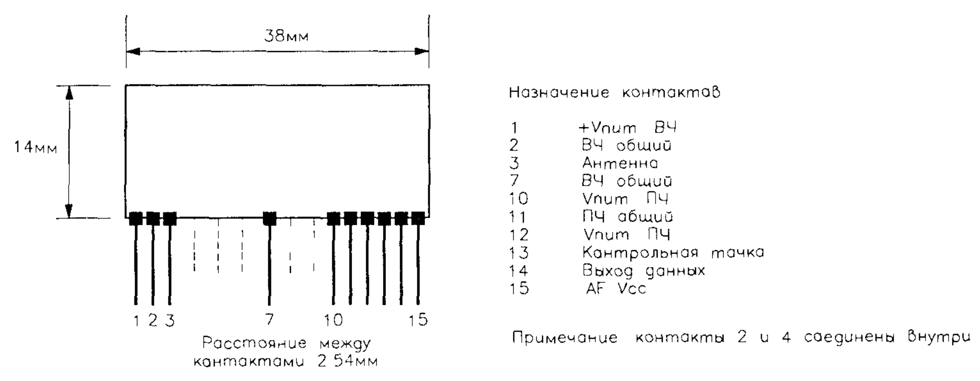


Рис. 8.8. Расположение выводов AM-HHR3

Рис. 8.9. Схема радиопередатчика цифровых данных

передатчиков TXM-418 или AM-TX1. В качестве параллельно-последовательного преобразователя применяется микросхема HT-12E. 12 бит данных могут поступать с экспериментальной платы параллельного порта. Адресные входы микросхемы A0 – A6 соединены с контактами D1 – D7 на плате, а входы данных D0 – D3 – с контактами C1 – C4 на плате. Вход $\overline{TE}$ соединен с контактом D8 экспериментальной платы, вход A7 – постоянно с «землей».

На рис. 8.10 изображена схема, которая принимает радиосигнал и трансформирует последовательные данные в параллельные. Используются приемные модули SILRX-418 и AM-HHR3 и преобразователь HT-12D. Такая система позволяет одному передатчику, соединенному с экспериментальной платой параллельного порта, передавать четыре бита данных одному из 127 приемников. Она может применяться в приложениях дистанционного управления. Пару приемник/передатчик допустимо также использовать в беспроводных системах обмена данными. В подобные устройства часто включают АЦП с последовательным вводом/выводом.

8.4. Модули приемопередатчиков

В разделе приводится описание миниатюрного приемопередатчика, а также даются общие рекомендации по организации пакетной радиосвязи.

8.4.1. Приемопередатчик BiM-418-F

Модуль приемопередатчика BiM-418-F имеет две модификации: BiM-418-F (рис. 8.11) и BiM-433-F (Radiometrix). Первая работает в диапазоне 418 МГц

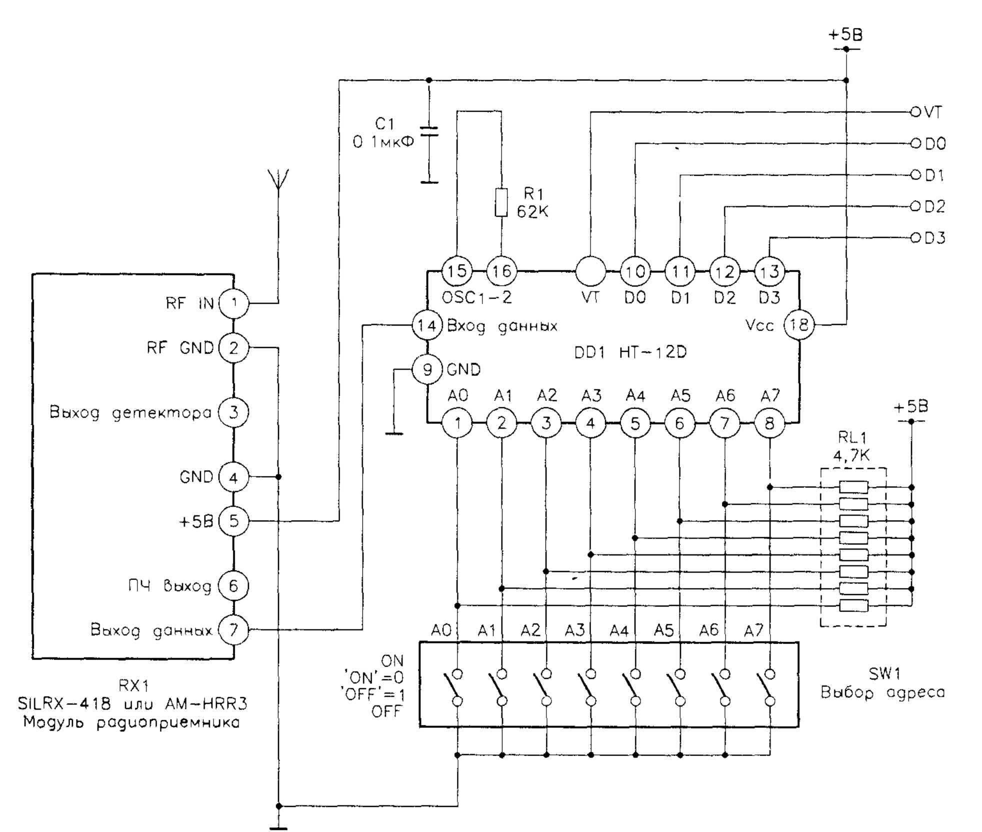


Рис. 8.10. Схема радиоприемника цифровых данных

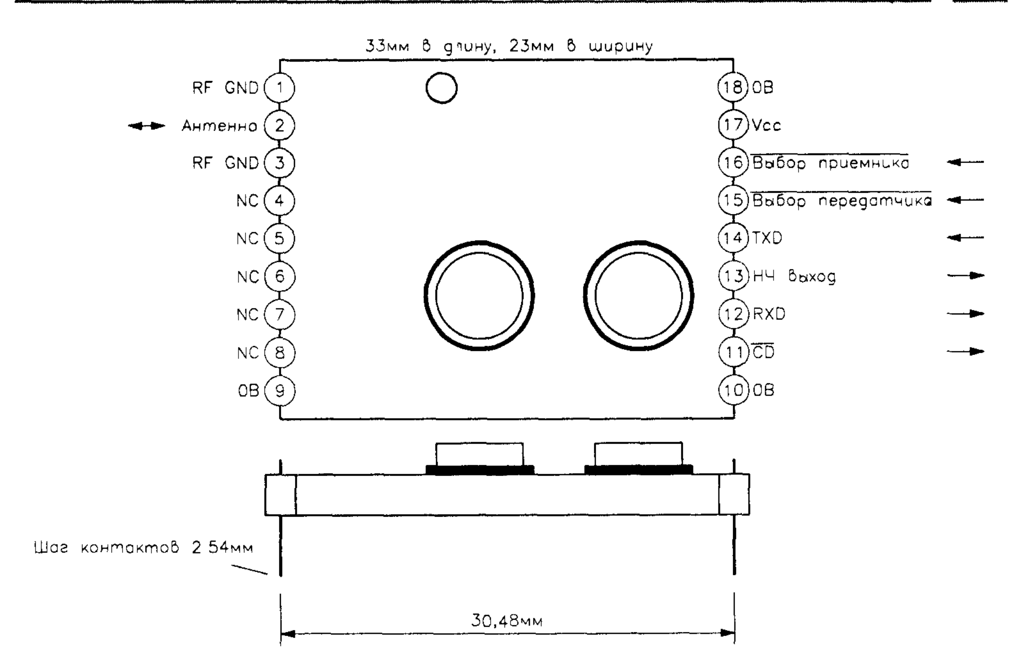


Рис. 8.11. *Назначение выводов приемапередатчика BiM-418-F*

и сертифицирована Сообществом по радиосвязи Великобритании (MPT1340). Вторая создана для Европы и функционирует в диапазоне 433,92 МГц. Обе модели способны организовать двунаправленную полудуплексную передачу данных со скоростью до 40 Кб/с на расстоянии до 30 м в здании и до 120 м на открытом пространстве.

Принцип работы блоков передатчика и приемника такой же, как у передатчика серии TMX и приемника SILRX, описанных выше. Контакты 9, 10 и 18 – это выводы заземления (0 В), подключенные к отрицательному проводу источника питания. Контакт 17 соединен с положительным проводом источника питания (Vcc). Необходим источник постоянного напряжения от 4,5 до 5,5 В. В режиме приема или передачи модуль потребляет ток 12 мА, в режиме ожидания – 1 мкА. Контакт 14 – вход передаваемых данных. Он может управляться непосредственно КМОП логикой, работающей от такого же напряжения питания, что и сам модуль. На этот контакт разрешается подавать аналоговые сигналы, генерируемые модемами или DTMF-кодерами. Контакт 12 – выход принятых данных, непосредственно соединенный с КМОП логикой. Контакт 13 – выход аналоговых сигналов, он подключается к модемам или DTMF-декодерам. Контакт 11 – вывод захвата несущей ($\overline{CD}$). Когда модуль функционирует в режиме приема, ноль на этом выходе означает, что уровень принимаемого сигнала выше установленного порога. Контакты 15 ($\overline{TX}$) и 16 ($\overline{RX}$) используются для установки режима работы модуля:

- контакт 15 = 1, контакт 16 = 1 – режим ожидания;
- контакт 15 = 1, контакт 16 = 0 – режим приема;

- контакт 15 = 0, контакт 16 = 1 – режим передачи;
- контакт 15 = 0, контакт 16 = 0 – режим циклического самотестирования.

Контакты 1 и 3 – это общий провод высокочастотной секции модуля. Они внутрисхемно соединены с контактами 9, 10 и 18 и с шасси корпуса, контакт 2 – с антенной. Для этих модулей рекомендуется использовать три типа антенн. Конфигурация антенн и таблица с их свойствами приведены на рис. 8.6.

8.4.2. Требования к передаваемым последовательным данным

Для качественной передачи данных необходимо соблюдать несколько ограничений. Длительность импульса (то есть время между двумя ближайшими фронтами) последовательного кода должна находиться в диапазоне от 25 мкс до 2 мс. Для приемника BiM перед передачей данных необходимо послать преамбулу 10101010 длительностью минимум 3 мс. Приемник настроен на получение высокочастотного сигнала со средним заполнением 50:50 и средней длительностью периода 4 мс, то есть количество нулевых и единичных битов за время 4 мс должно быть примерно одинаковым. Данное соотношение допускается изменять до 30:70 или 70:30, однако это может привести к снижению помехоустойчивости и увеличению ошибок при приеме.

Модули приемопередатчиков используются для передачи сигналов в формате протокола RS232 между компьютерами. Экспериментальная схема приведена на рис. 8.12. Обмен данными в последовательном формате осуществляется со скоростью от 4,8 до 38,3 Кб/с.

Для того чтобы передаваемые данные удовлетворяли требованиям BiM, их необходимо пакетировать. *Пакетированные данные* состоят из следующих частей:

- 3 мс преамбула (55h или AAh) для подстройки приемника BiM;
- один или два байта FFh;
- 1 байт 01h в качестве флага начала передачи;
- байты данных;
- биты контрольной суммы.

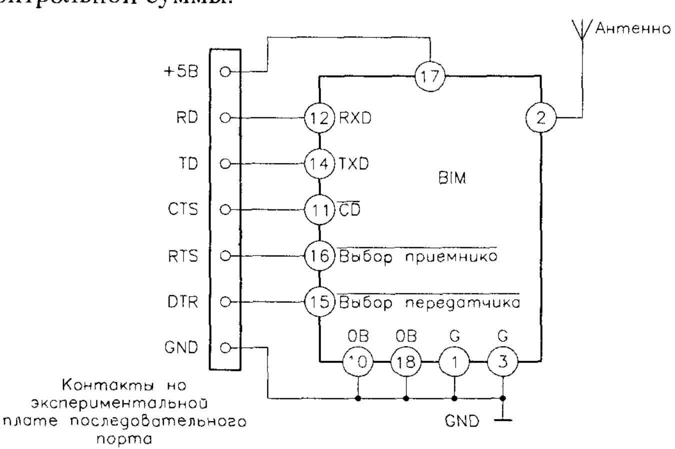


Рис. 8.12. Схема радиомодема

На практике формат пакета зависит от конкретной реализации.

Существует три способа обеспечения среднего коэффициента заполнения 50:50. Первый способ заключается в делении каждого байта пополам. Первая половина – это передаваемые биты данных, вторая – их дополнение до 1. При передаче каждого байта гарантированный коэффициент заполнения составляет 50:50. Среди 256 возможных комбинаций восьмиразрядного кода 70 комбинаций состоят из четырех единиц и четырех нулей и при этом имеют коэффициент заполнения 50:50. Например, 17h, 1Bh, 27h, E8h и т.д. Их можно передать между двумя последовательными портами в формате RS232 с одним стартовым битом, одним стоповым битом и без проверки на четность. Реальную информацию допустимо закодировать, используя только указанные комбинации. В этом состоит второй способ. По третьему способу каждый байт передается дважды. Первый байт содержит фактические данные, а второй – дополнение первого до 1 в каждом разряде. Коэффициент заполнения составляет 50:50.

8.5. Модем для работы в бытовой электросети LM1893

Микросхема LM1893 (National Semiconductors) – это двухканальный приемопередатчик, разработанный специально для обмена данными между удаленными терминалами с использованием бытовой электросети в качестве канала связи на расстояния в пределах одной подстанции. Расположение выводов микросхемы показано на рис. 8.13, типовое включение – на рис. 8.14.

Напряжение питания должно быть от 14 до 30 В. Vcc (контакт 15) и GND (контакт 14) соединены с положительным и отрицательным проводами источника питания. Режимы приема (Rx) и передачи (Tx) выбираются с помощью входа Tx/Rx (контакт 5). Если Tx/Rx = 1, то микросхема работает в режиме передачи (Tx). Входные данные со скоростью до 5 Кб/с подаются на вход DATA IN (контакт 17). Несущая частота находится в диапазоне 50–300 кГц и определяется элементами R3, R1 и C8. Сигнал подается на формирователь синусоидального колебания, затем через схему автоматической регулировки усиления (АРУ) поступает на выходной усилитель тока. Схема АРУ обеспечивает постоянный уровень сигнала на выходе при изменении волнового сопротивления электросети.



Рис. 8.13. Расположение выводов LM1893

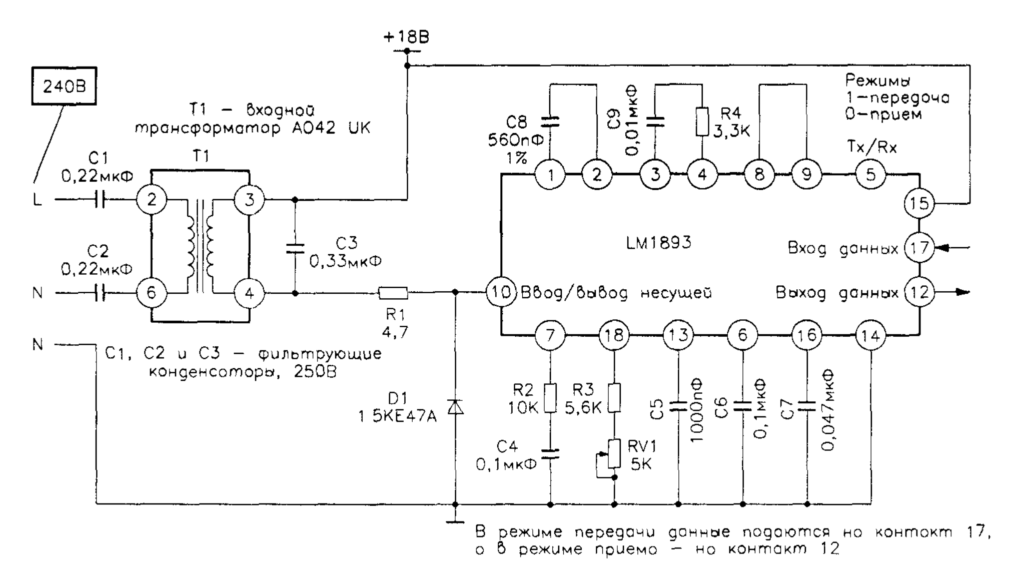


Рис. 8.14. Схема включения микросхемы LM1893

Элементы C4 и R2 управляют динамическими характеристиками схемы АРУ. Напряжение возбуждения с контакта 10 (выход при работе в режиме передачи и вход в режиме приема) вызывает резонанс напряжений в колебательном контуре, образованном первичной обмоткой трансформатора T1 и конденсатором C3. Конденсатор C3 подбирается таким образом, чтобы резонансная частота колебательного контура была равна частоте несущей. Далее с вторичной обмотки этого трансформатора информация поступает в электропроводку. Элементы R1 и D1 используются для защиты микросхемы от кратковременных бросков напряжения, которые в бытовой сети происходят достаточно часто.

Если Tx/Rx = 0 (контакт 5), то микросхема работает как приемник. Блок передатчика при этом отключен. Сигнал, поступающий на вход схемы, представляет собой смесь различных сигналов, присутствующих в сети питания. Входной сигнал подается на входной фильтр верхних частот, состоящий из конденсаторов C1 и C2. Колебательный контур – это полосовой фильтр, который также подавляет помехи. Описанные меры позволяют значительно ослабить напряжение сети 240 В и импульсные помехи. Затем сигнал поступает на вход микросхемы (контакт 10). После преобразований и детектирования принятая информация в последовательном двоичном коде подается на выход с открытым коллектором (контакт 12). Параметры приемной части микросхемы определяются конденсаторами C5, C6 и C7.

8.6. Интерфейс RS485

Интерфейс RS485 представляет собой улучшенную версию интерфейса RS232. Он широко используется при разработке систем управления и передачи данных.

Для улучшения качества передачи данных и уменьшения шума в канале связи применяется шина, выполненная в виде витой пары. Максимальная скорость передачи данных составляет 10 Мб/с, максимальное расстояние – 1200 м.

В основе RS485 лежит приемник с дифференциальным входом для подавления синфазных помех. Передача данных через этот интерфейс защищена лучше, поскольку канал связи имеет более стабильные характеристики, чем интерфейс RS232. Подключение передатчиков и приемников к каналу связи согласовано. На практике величина волнового сопротивления канала связи равна величине выходного сопротивления передатчика.

Микросхема SN75176B (Texas Instruments, RS630-904) – это устройство, позволяющее достаточно просто организовать обмен данными в соответствии со спецификацией на интерфейс RS485 (рис. 8.15). Одно устройство управления RS485 может контролировать 32 приемника. Все передатчики и приемники подсоединены к шине двумя проводами, режим работы шины полудуплексный: два и более передатчика одновременно функционировать не могут. Остальные либо имеют высокое сопротивление, либо работают в режиме приема. На рис. 8.16 изображена простая локальная сеть с использованием протокола RS485 и экспериментальной платы последовательного порта.

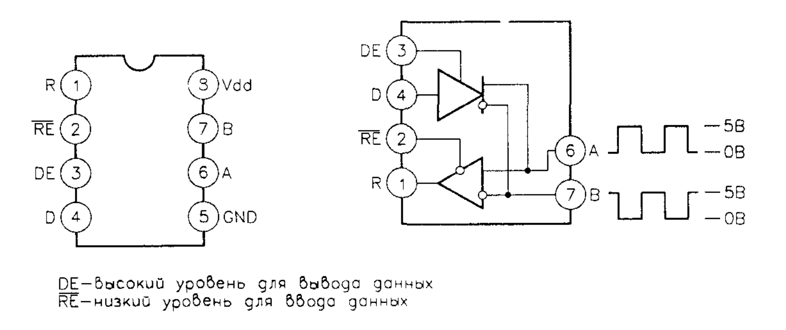


Рис. 8.15. Расположение выводов и внутренняя блок-схема SN75176B

8.7. Инфракрасные линии передачи данных

IrDA (Infra-Red Data Association) – это Ассоциация передачи данных в инфракрасном диапазоне, объединяющая несколько фирм-производителей в области технологии оптической передачи данных. Ее цель – разработка стандарта для обмена данными с помощью ИК излучения. Такая технология применяется при организации обмена информацией между настольными/портативными компьютерами и принтерами, телефонными и факсимильными аппаратами. Большое преимущество обмена данными в инфракрасном диапазоне – полное отсутствие кабелей связи между устройствами.

Технология IrDA подразумевает работу на относительно небольших расстояниях: так можно уменьшить потребляемую мощность и устранить взаимовлияние различных устройств. Угол диаграммы направленности равен 30°. ИК диоды излучают сигнал в диапазоне 850–900 нм. Стандарт IrDA-1 поддерживал скорость

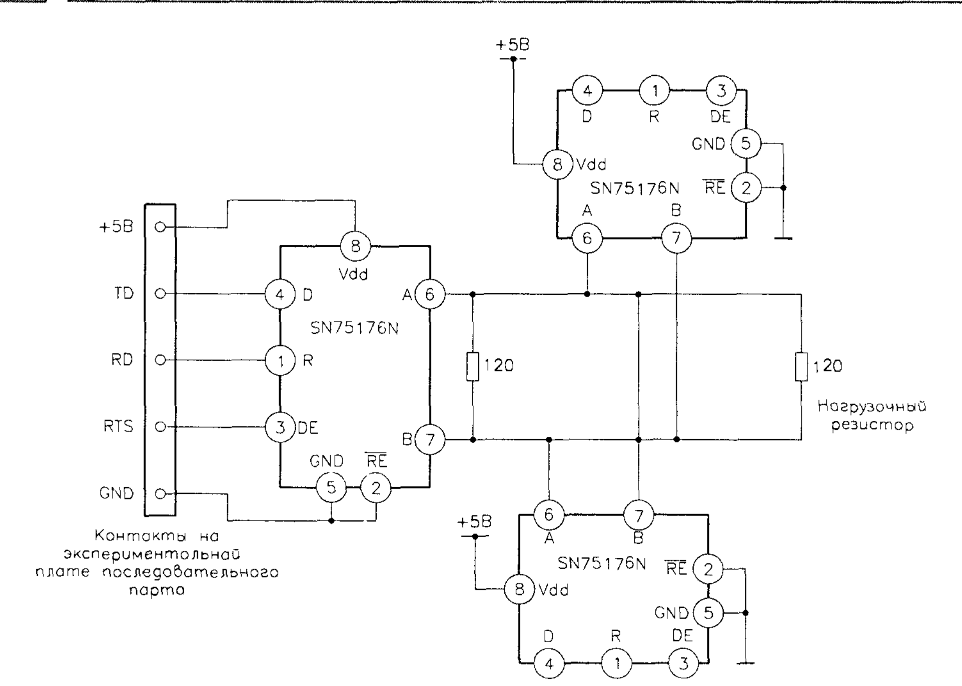


Рис. 8.16. Схема преобразования протокола RS232 в протокол RS485

передачи данных до 115,2 Кб/с в полудуплексном режиме, затем был принят стандарт со скоростями до 4 Мб/с.

Протокол IrDA – это продолжение RS232 (или UART). Светодиод соединяется с последовательным выходом RS232 через передающий ИК интерфейс, фотодиод – с приемником через приемный ИК интерфейс. Излучающий интерфейс уменьшает длительность импульсов RS232 максимум до 3/16 исходной, что сокращает потребляемую светодиодом мощность. На приемном конце ИК интерфейс восстанавливает исходную длительность импульсов для их нормальной обработки протоколом RS232.

Существует несколько разновидностей модулей для приложений IrDA. Они состоят из светодиода, устройства управления, фотодиода и усилителя. В качестве примера можно привести микросхему HSDL-1000-101 (Hewlett Packard, RS193-4780) – см. рис. 8.17.

На вход модуля поступают укороченные последовательные импульсы. С помощью светодиода они преобразуются в световые сигналы. Световые инфракрасные импульсы, принимаемые фотодиодом, преобразуются в сигналы с уровнями ТТЛ. Приемный и передающий ИК интерфейсы, изменяющие их длительность, не входят в состав модуля.

Для сопряжения интерфейсов IrDA и RS232 можно использовать приемный и передающий ИК интерфейс HSDL-7000 (Hewlett Packard, RS233-2242), как показано на рис. 8.18.

Назначение контактов

1 Конденсатор компенсации дневного света
2 Конденсатор развязки
3 Uпит
4 Выход принятых данных
5 Общий
6 Вход данных для передачи
7 Катод светодиода
8 Анод светодиода

1 2 3 4 5 6 7 8

+5В
Rled 8
LED C
LED A
7
8
LED
Rl 300
TXD
6
Vpin
Фотодиод
Компаратор
RXD
4
1
Vcc
3
Vpin
CX2
Vpin
2
GND
5
Vcc
GND
CX4
CX3
CX1

CX1 конденсатор компенсации дневного света, 0,22мкФ
CX2 конденсатор развязки, 0,4мкФ
CX3 0,1мкФ
CX4 4 7мкФ

Рис. 8.17. Расположение выводов и внутренняя блок-схема HSDL-1000

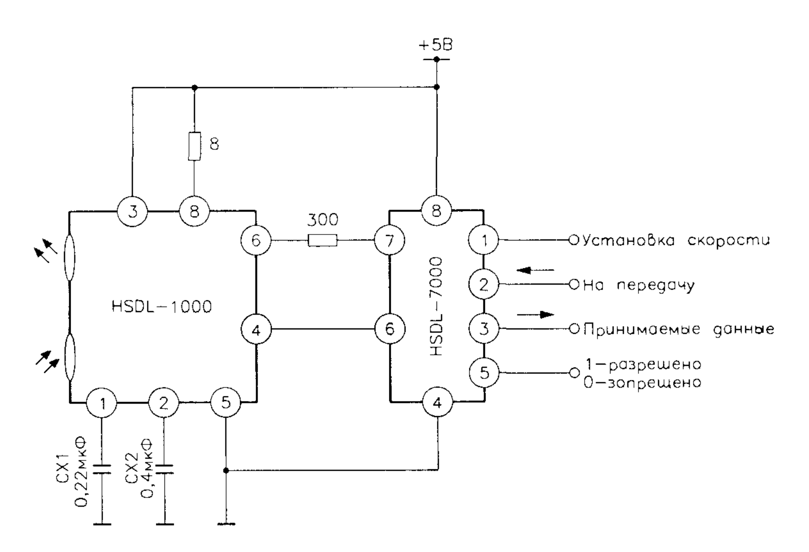


Рис. 8.18. Схема для передачи данных по оптическому каналу связи

Микросхема CS8130 (Crystal Semiconductor, RS207-2473) – это инфракрасный приемопередатчик (рис. 8.19). Он принимает данные от микросхемы UART со скоростями от 1200 до 115200 бод. Для работы микросхемы необходимы внешний фотодиод и светодиод. Напряжение источника питания от 2,7 до 5,5 В, потребляемый ток 2,5 мА.

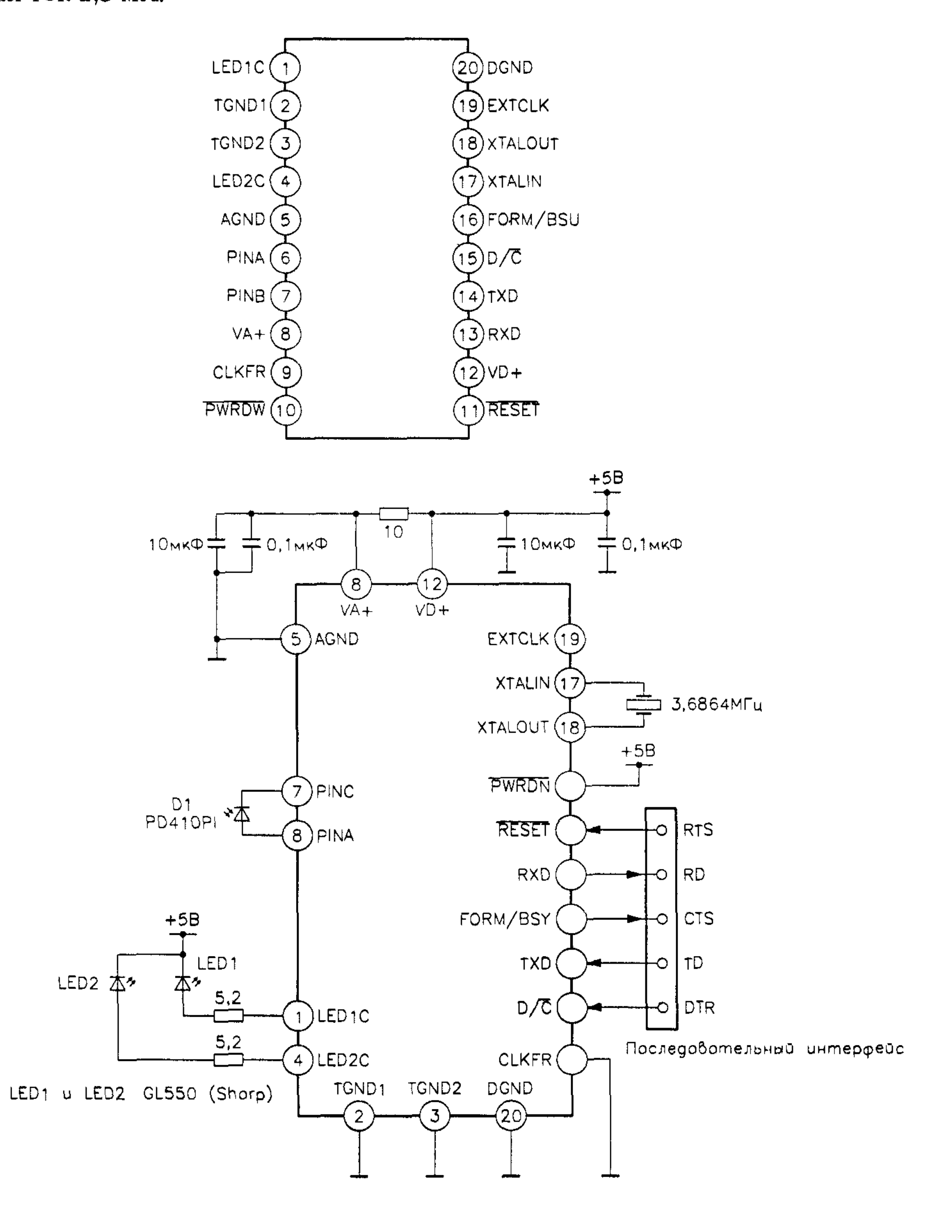


Рис. 8.19. Расположение выводов и типовое включение микросхемы CS8130

Микросхема имеет четыре режима передачи: IrDA, режим амплитудной манипуляции (АМ) на частоте 500 кГц, режим дистанционного управления на частоте 38 кГц и режим непосредственного доступа. В режиме IrDA ИК излучение соответствует логическому 0, отсутствие излучения – логической 1.

Длительность импульса составляет от 1,6 (для скорости 115200) до 78 мкс (для скорости 1200). Кроме того, может использоваться фиксированная длительность 1,6 мкс для всех скоростей. Исходная скорость для режима IrDA равна 9600 бод, но разрешается установить скорость от 1200 до 115200 бод. В режиме АМ присутствие несущего колебания на частоте 500 кГц соответствует логическому 0, отсутствие несущей – логической 1. Допустимы скорости 9600, 19200 и 38400 бод. Режим дистанционного управления аналогичен режиму АМ, за исключением частоты несущей, которая равна 38 кГц. Этот режим, как правило, применяется для дистанционного управления телевизионными приемниками. В режиме непосредственного доступа ИК передатчик отображает то, что имеется на входе TXD. Логическая 1 соответствует выключенному светодиоду, логический 0 – включенному. На приемном конце высокий уровень на выходе RXD означает, что световой энергии не обнаружено, а низкий – что был принят световой импульс.

При передаче данные сначала записываются в микросхему через вход TXD, а затем передаются с помощью выбранного способа модуляции. Режим передачи выбирается путем записи управляющего слова в соответствующий внутренний регистр управления. Для изменения режимов работы в микросхеме имеются различные регистры управления. Режимы приема также выбираются посредством записи управляющего слова во внутренние регистры. Данные записываются в регистры управления при подаче на вывод $D/\overline{C}$ (контакт 15) сигнала низкого уровня. Детальное описание микросхемы приведено в документации изготовителя.

СПИСОК ЛИТЕРАТУРЫ

1. Owen Bishop. Easy Add-on Projects for Spectrum, ZX81 and Ace, ISBN 0859340996. Bernard Babani Publishing Ltd., 1983.
2. Roger G. Gilbertson. Muscle Wires Project Book, ISBN 1-879896133. Mondotronics, Inc., 1994.
3. Hans-Peter Messmer. The Indispensable PC Hardware Book, ISBN 0201624249. Addison-Wesley, 1993.
4. Mustafa A. Mustafa. Microcomputer Interfacing and Applications, Second edition, ISBN 0750617527. Butterworth-Heinemann, 1994.
5. Que Corporation. Using Visual Basic 3, ISBN 156529763X, 1995.
6. Michael Tooley. Electronic Circuits Handbook, ISBN 0434919683. Butterworth-Heinemann, 1990.
7. Data sheets for components from various manufacturers.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

И

К

Л

М

О

П

Пей Ан

Сопряжение ПК с внешними устройствами

Главный редактор *Захаров И. М.*
Научный редактор *Мацонашвили М. А.*
Выпускающий редактор *Левицкая Т. В.*
Технический редактор *Александрова О. С.*
Графика *Бахарев А. А.*
Дизайн обложки *Панкусова Е. Н.*

ИД № 01903 от 30.05.2000

Подписано в печать 30.04.2001. Формат 70×100[1]/[16].
Гарнитура «Петербург». Печать офсетная.
Усл. печ. л. 20. Тираж 3000 экз. Зак. № 199.

Издательство «ДМК Пресс», 105023, Москва, пл. Журавлева, д. 2/8.
Электронные адреса: www.dmkpress.ru, info@dmk.ru

Отпечатано на ордена Трудового Красного Знамени
ГУП Чеховский полиграфический комбинат
Министерства Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций
142300, г. Чехов Московской области
Тел. (272) 71-336. Факс (272) 62-536